
Forecasting in Offline Reinforcement Learning for Non-stationary Environments: Supplementary Material

Suzan Ece Ada^{1,2} Georg Martius² Emre Ugur¹ Erhan Oztop^{3,4}

¹Bogazici University, Türkiye ²University of Tübingen, Germany

³Ozyegin University, Türkiye ⁴Osaka University, Japan

ece.ada@bogazici.edu.tr

A Related Work: Continued

Offline Reinforcement Learning A high-level overview of existing work in offline reinforcement learning (RL) identifies three predominant strategies: policy constraint methods [14, 43, 44], pessimistic value function methods which assign low values to OOD actions [45], and model-based offline RL methods. Policy constraint methods actively avoid querying OOD actions during training by leveraging probabilistic metrics which can be explicit [46, 47], implicit [48, 49] f -divergence [50], or integral probability metrics [51]. These metrics ensure that the learned policy π_θ remains close to the behavior policy π_β that generated the offline RL dataset [1]. Similarly, pessimistic value function approaches regularize the value function or the Q-function to avoid overestimations in OOD regions. Model-based offline RL methods [52], on the other hand, focus on learning the environment’s dynamics, benefiting from the strengths of supervised learning approaches. However, in the same vein, these methods are susceptible to the distribution shift problem in Offline RL [1]. We detail the offline RL algorithms used in our experiments in Section E.

Park et al. [53] emphasize the challenges of generalizing policies to test-time states which are not in the support of the offline RL dataset. While prior works have investigated the issue of generalization [38, 54], in testing-time robust RL methods [3] this challenge is exacerbated through the introduction of noise into the states by an unknown adversary.

B Background

B.1 Reinforcement Learning

Markov Decision Processes (MDPs) are often used to formalize Reinforcement Learning (RL). MDP is defined by the tuple $\mathcal{M} \doteq (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0, \gamma)$ where \mathcal{S} is the state space, \mathcal{A} is the action space, \mathcal{T} is the transition function (which may be deterministic or stochastic), $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, ρ_0 is the initial state distribution and $\gamma \in [0, 1)$ is the discount factor [55]. In online and off-policy RL algorithms an agent can interact with the environment using a parameterized policy $\pi_\theta(\mathbf{a}|\mathbf{s})$, to maximize the expected return $\mathbb{E}_\pi [\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$. In contrast, offline RL requires the agent to learn from a static dataset $\mathcal{D} = \{(\mathbf{s}_k, \mathbf{a}_k, \mathbf{s}'_k, r_k)\}_{k=1}^N$ generated by a generally unknown behavior policy $\pi_\beta(\mathbf{a}|\mathbf{s})$ [1].

B.2 Non-Stationary Environments

We next review tangential definitions and formalisms used for non-stationary environments.

Definition B.1 (Partially Observable Markov Decision Processes [6, 7, 16]). A Partially Observable Markov Decision Process (POMDP) is given by the tuple

$$\hat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \mathbf{x}, \rho_0, \gamma),$$

where \mathcal{O} is the observation space and \mathbf{x} is the observation function. This function can be deterministic ($\mathbf{x} : \mathcal{S} \rightarrow \mathcal{O}$) [6, 17] or stochastic ($\mathbf{x}(o | s)$) for $o \in \mathcal{O}, s \in \mathcal{S}$ [7].

To capture a broader range of non-stationary RL scenarios, Khetarpal et al. [6] present a *general non-stationary RL* formulation, which allows each component of the underlying MDP or POMDP to evolve over time. Concretely, this time-varying tuple is denoted as $(\mathcal{S}(t), \mathcal{A}(t), \mathcal{T}(t), \mathcal{R}(t), \mathbf{x}(t), \mathcal{O}(t))$ where each element is represented by a function $\varphi(i, t)$, indicating its variation with time t and input i [6]. *Scope*, denoted by a set κ , specifies which of these components vary.

Definition B.2 (Scope of Non-Stationarity [6]). Given the general non-stationary RL framework, the *scope of non-stationarity* is the subset

$$\kappa \subseteq \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathbf{x}, \mathcal{O}\},$$

indicating which components of the environment evolve over time.

Notably, Chandak [7] defines Non-Stationary Decision Process (NSDP) as a sequence of POMDPs, with non-stationarity confined to the subset $\kappa \subseteq \{\mathcal{R}, \mathcal{T}, \mathbf{x}\}$, where the initial state distribution $\rho_{0,j}$ varies between POMDPs. Existing research in non-stationary RL largely focuses on the episodic evolution of transition dynamics and reward functions [27]. In contrast, the evolution of observation functions remains underexplored, despite its potential for real-world applicability and thus demands further investigation [6]. Handling inaccurate state information is crucial in non-stationary RL since an agent may encounter non-stationarity due to its own imperfect perception of the state while the underlying physics of the environment remains unchanged [6].

B.3 Diffusion Models

Diffusion models [12] aim to model the data distribution with $p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$ from samples \mathbf{x}_0 in the dataset. The joint distribution $p_\theta(\mathbf{x}_{0:T})$, is modeled as a Markov chain, where $\mathbf{x}_1, \dots, \mathbf{x}_T$ are the latent variables with the same dimensionality as the data samples. The joint distribution is given by

$$p_\theta(\mathbf{x}_{0:T}) := \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) \quad (6)$$

where $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$. The forward process $q(\mathbf{x}_{1:T} | \mathbf{x}_0) := \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1})$ involves adding a small amount of Gaussian noise to the data sample at each diffusion timestep to obtain the latent variables following a variance schedule $\{\beta_t = 1 - \alpha_t \in (0, 1)\}_{t=1}^T$. Here, the encoder transitions are $q(\mathbf{x}_t | \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t} \mathbf{x}_{t-1}, (1 - \alpha_t) \mathbf{I})$ [11]. Assuming we have access to the true data sample during training, using recursion and the reparameterization trick, we can obtain samples \mathbf{x}_t at any timestep t in closed form with $q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$ where $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ [11]. During training, the evidence lower bound is oftentimes maximized by a simplified surrogate objective [11]. After learning the parameters of the reverse process, we can sample \mathbf{x}_T from $\mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$ to start generating samples through an iterative denoising procedure.

C Details on FORL

C.1 Conditional Diffusion Model Details

Our aim is to learn the reverse diffusion process, by modeling $p_\theta(\mathbf{s}_t^{(n-1)} | \mathbf{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)})$ as a Gaussian distribution $\mathcal{N}(\mathbf{s}_t^{(n-1)}; \boldsymbol{\mu}_\theta(\mathbf{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n), \boldsymbol{\Sigma}_\theta(\mathbf{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n))$ where n is the diffusion timestep and t is the RL timestep. We approximate this mean $\boldsymbol{\mu}_\theta(\mathbf{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n)$ using a conditional noise model ϵ_θ with

$$\frac{\mathbf{s}_t^{(n)}}{\sqrt{\alpha(n)}} - \frac{1 - \alpha(n)}{\sqrt{1 - \bar{\alpha}(n)} \sqrt{\alpha(n)}} \epsilon_\theta(\mathbf{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n)$$

Algorithm 2 Training

Require: Offline RL dataset \mathcal{D} **Initialize:** ϵ_θ

- 1: **for** each iteration **do**
 - 2: $\{(\tau_{(t,w)}, s_t)\} \sim \mathcal{D}$
 - 3: $n \sim \mathcal{U}(\{1, 2, \dots, N\})$
 - 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 5: Take gradient descent step on
 $\nabla_\theta \left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_{(n)}} s_t + \sqrt{1 - \bar{\alpha}_{(n)}} \epsilon, \tau_{(t,w)}, n \right) \right\|^2$
 - 6: **end for**
 - 7: **return** ϵ_θ
-

and fix the covariance $\Sigma_\theta(s_t^{(n)}, \tau_{(t,w)}, n)$ with $(1 - \alpha_{(n)})\mathbf{I}$ [11]. Selecting a large number of diffusion timesteps can significantly increase the computational complexity of our algorithm. Hence, we use variance preserving stochastic differential equations (SDE) [18] following the formulation in [19] $\alpha_{(n)} = e^{-(\beta_{\min}(\frac{1}{N}) + (\beta_{\max} - \beta_{\min})\frac{2n-1}{2N^2})}$ where $\beta_{\max} = 10$ and $\beta_{\min} = 0.1$. We use the noise prediction model [11] with reverse diffusion chain

$$s_t^{(n-1)} | s_t^{(n)} = \frac{s_t^{(n)}}{\sqrt{\alpha_{(n)}}} - \frac{1 - \alpha_{(n)}}{\sqrt{\alpha_{(n)}(1 - \bar{\alpha}_{(n)})}} \epsilon_\theta(s_t^{(n)}, \tau_{(t,w)}, n) + \sqrt{1 - \alpha_{(n)}} \epsilon \quad (7)$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $n = N, \dots, 1$, and $\epsilon = 0$ for $n = 1$ [11].

By using the conditional version of the simplified surrogate objective from [11] we minimize the FORL model loss $\mathcal{L}_p(\theta)$, with Eq. (4) in the main paper. We can train the noise prediction model by sampling $s_t^{(n)}$ for any diffusion timestep in the forward diffusion process, utilizing reparametrization and recursion [11]. We use the true data sample s_t from the offline RL dataset to obtain the noisy state $s_t^{(n)} = \sqrt{\bar{\alpha}_{(n)}} s_t + \sqrt{1 - \bar{\alpha}_{(n)}} \epsilon$. Leveraging our model’s capacity to learn multimodal distributions, we generate a set of k samples (predicted state candidates) $\{s_t^{(0)}\}$ in parallel from the reverse diffusion chain using Eq. (5).

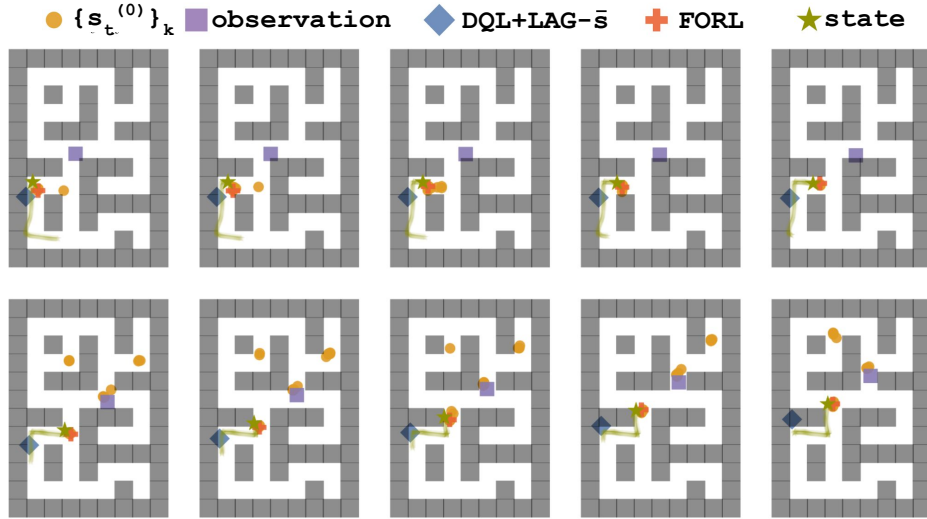


Figure 12: As the agent navigates in the maze2d-large environment [15], illustrations of states; predicted states from FORL (ours); FORL diffusion model predictions $\{s_t^{(0)}\}_k$; observations; and predicted states from DQL+LAG- \bar{s} are shown. Environment timesteps progress from left to right.

D Baselines

In this section we explain the baselines used in Table 1 and Figure 14. Diffusion Q-Learning (DQL) [14] serves as our base policy across all baselines and for our method in D4RL experiments [15]. Details on DQL are provided in Section E.

D.1 DQL Extensions with *Zero-Shot FM*

For our baselines, we use a diffusion-based offline RL policy, DQL [14], with three variations:

- **DQL:** DQL policy directly generates actions conditioned on the observations. We include it to demonstrate how large episodic offsets degrade policy performance, and its consistently poor performance underscores the difficulty introduced by our offset settings.
- **DQL+LAG- \bar{s} :** We use *Zero-Shot FM* (Lag-Llama [10]) to forecast episodic offsets over a horizon of P episodes using pre-deployment offset history, and then subtract the mean predicted offset corresponding to that episode from the observations. Thus, DQL generates actions conditioned on the sample mean of the forecasted states $\{\hat{s}_t^b\}_l$. Although DQL+LAG- \bar{s} outperforms DQL, a straightforward application of forecasting and decision-making reveals performance degradation when offsets lack adaptive correction.
- **DQL+LAG- \tilde{s} :** We use the sample median instead of the sample mean in DQL+LAG- \bar{s} , to mitigate outlier effects. As shown in Figure 14, median-based bias removal yields no significant improvement over the mean-based approach.

D.2 DMBP+LAG

We extend the Diffusion Model-Based Predictor (DMBP) [3], a model-based, robust offline RL method trained on unperturbed D4RL benchmarks, to correct test-time state observation perturbations. DMBP has proven robust under Gaussian noise (varying σ), uniform noise, and adversarial perturbations—including maximum action-difference (MAD) and minimum Q-value (Min-Q) attacks. Similar to our framework, perturbed states are passed to DMBP, which removes the perturbations; the generated state is then fed to the policy. For a fair comparison, we adopt the policies also used in their work for our experiments, RORL [25] and TD3BC [2] alongside DQL.

DMBP+LAG extends DMBP with the forecasting module in our experiments. At each timestep, we remove offset biases using the sample mean of forecasted states $\{\hat{s}_t^b\}_l$ (as in DQL+LAG- \bar{s}) before DMBP correction. DMBP+LAG thus sequentially applies forecast-based offset compensation, followed by model based perturbation removal, and then queries the policy. DQL [14] remains the shared policy. A naive integration of DMBP relies on initial ground-truth states and underperforms.

For the evaluation of DMBP we use the open source code and the suggested hyperparameters available in [56]. To improve the performance of DMBP, we conducted evaluations at test-time and reported the best-performing results for a range of different diffusion timesteps. DMBP requires the diffusion timesteps to be manually defined based on different noise scales and types. Hence, we use the diffusion timesteps across a range of noise scales $\{0.15, 0.25, 0.5\}$ for maze and $\{0.05, 0.1, 0.15, 0.25\}$ for kitchen environments and identify that the best performance requires different noise scales across environment datasets. In particular, we report the best performing noise scale of 0.15 for the maze2d-medium environment, 0.25 for the maze2d-large environment, 0.5 for the antmaze-umaze-diverse environment, 0.15 for the antmaze-medium-diverse environment, 0.25 for the antmaze-large-diverse environment, 0.05 for kitchen-complete.

E Offline Reinforcement Algorithms

DQL [14] DQL [14] is an offline RL algorithm that uses policy regularization via a conditional diffusion model [11]. Wang et al. [14] shows that Gaussian policies lack the expressiveness needed to capture the possibly multimodal and skewed behavior policy in offline datasets, which, in turn, limits performance. To remedy this, DQL uses a conditional diffusion model for the behavior-cloning term, shown as the first part of Equation 8 based on a state-conditioned version of the simplified Denoising Diffusion Probabilistic Models (DDPM) objective [11]. To steer action generation toward high-reward

regions, the policy improvement loss also includes Q-value guidance, shown as the second term below [14]

$$\mathbb{E}_{n \sim \mathcal{U}, (s, a) \sim \mathcal{D}, \epsilon \sim \mathcal{N}(0, I)} \left[\left\| \epsilon - \epsilon_\phi \left(\sqrt{\bar{\alpha}(n)} a + \sqrt{1 - \bar{\alpha}(n)} \epsilon, s, n \right) \right\|^2 \right] - \alpha \mathbb{E}_{s \sim \mathcal{D}, a^0 \sim \pi_\phi} [Q_\psi(s, a^0)]. \quad (8)$$

Here, a reverse diffusion process conditioned on state s , denoted by $\pi_\phi(a | s)$, represents the policy. The Q-networks are trained using the double Q-learning trick [57] and Bellman operator minimization [43, 58], with [14]

$$\mathbb{E}_{(s_t, \mathbf{a}_t, s_{t+1}) \sim \mathcal{D}, \mathbf{a}_{t+1}^0 \sim \pi_{\phi'}} \left[\left\| \left(r(s, \mathbf{a}) + \gamma \min_{i=1,2} Q_{\psi'_i}(s_{t+1}, \mathbf{a}_{t+1}^0) \right) - Q_{\psi_i}(s_t, \mathbf{a}_t) \right\|^2 \right] \quad (9)$$

where \mathbf{a}_{t+1}^0 is sampled from the diffusion policy conditioned on s_{t+1} , and $Q_{\psi_i}, Q_{\psi'_i}, \phi'$ denote the critic and target-critic networks, target policy network, respectively.

TD3BC [2] TD3BC [2] extends the Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [59] to the offline RL setup. TD3BC incorporates a behavior cloning regularization term, normalizes state features within the offline RL dataset, and scales the Q-function using a hyperparameter with an added normalization term. TD3BC is a straightforward yet effective method that is also computationally efficient. The results in Table 2 and Table 3 show that although extending TD3BC with the forecasting module (TD3BC+LAG- \bar{s}) and a combination of the forecasting module and a diffusion model-based predictor (DMBP+LAG-T) improve performance, FORL-T achieves better performance across a diverse range of non-stationarities.

RORL [25] RORL [25] addresses adversarial perturbations of the observation function by learning a conservative policy that aims to be robust to out-of-distribution (OOD) state and action pairs. To achieve this, it introduces a conservative smoothing mechanism that balances mitigating abrupt changes in the value function for proximate states and avoiding value overestimation in risky regions that are absent from the dataset. Concretely, RORL regularizes both the policy and the value function, leveraging bootstrapping Q-functions and conservative smoothing of the perturbed states. This formulation yields robust training under adversarial perturbations in the observation function while preserving strong performance even in unperturbed environments. However, although using RORL as our base policy improved performance in maze2d environments, FORL significantly outperforms both its naive usage, the extension with our forecasting module RORL+LAG- \bar{s} , and DMBP+LAG-R in Table 2 and Table 3. These results demonstrate that policies designed to be robust to sensor noise or adversarial attacks fail to cope with evolving observation functions that introduce non-stationarity into the environment.

IQL [26] Implicit Q-Learning [26] first learns a value function by expectile loss and a Q-function by Mean Squared Error (MSE) Loss without using the policy and instead using actions from the dataset. In doing so, they avoid approximating the values of unseen actions. Then, it learns the policy using advantage weighted regression [48, 49, 60, 61] using the learned Q-function and value function. We use the open-source implementation of IQL from [62] which references the source [63]. Results in Table 4 show FORL-I outperforms the baselines IQL, IQL+LAG- \bar{s} , and DMBP+LAG-I across all environments.

FQL [23] Flow Q-learning (FQL) [23] is a recent offline RL policy that shows strong performance on the OGBench [21]. We use FQL for the offline RL environments in OGBench and adopt the hyperparameters from the open-source implementation¹. Similar to DQL, which uses diffusion models, FQL can learn an expressive policy. FQL trains two policies: (i) a flow policy trained with flow matching on the offline RL dataset for behavior cloning conditioned on the state, and (ii) a one-step policy trained with a distillation loss using the flow policy [64–68] and a critic loss. This approach avoids expensive backpropagation through time; thus, it is fast during inference and training. Results in Tables 1 and 7 show FORL (also referred to as FORL (DCM)) outperforms the baselines when all baselines use FQL policy.

¹<https://github.com/seohongpark/fql/>

Table 3: **Normalized scores (mean \pm std.) for FORL and baselines with TD3BC and RORL on maze2d-medium.** Algorithms are grouped by their underlying policies—TD3 with Behavior Cloning (TD3+BC) [2] and Robust Offline Reinforcement Learning (RORL) [25] to highlight that performance variations stem from the algorithms themselves rather than the policies employed. Suffixes -T and -R denote the use of TD3+BC and RORL policies, respectively.

maze2d-medium	Td3Bc Policy				RORL Policy			
	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	37.4 \pm 9.5	16.2 \pm 5.1	16.2 \pm 7.3	22.1 \pm 6.6	80.7 \pm 14.2	57.9 \pm 8.6	47.8 \pm 10.2	52.7 \pm 5.0
real-data-B	3.6 \pm 2.3	6.1 \pm 4.4	14.4 \pm 8.1	28.6 \pm 19.0	37.8 \pm 7.5	85.9 \pm 19.8	91.0 \pm 21.6	109.6 \pm 19.5
real-data-C	-2.3 \pm 1.3	30.0 \pm 10.0	19.3 \pm 6.2	24.5 \pm 10.2	33.7 \pm 4.6	89.2 \pm 15.8	93.4 \pm 16.3	125.4 \pm 14.5
real-data-D	6.3 \pm 3.4	15.5 \pm 3.8	12.2 \pm 2.6	38.7 \pm 13.4	37.0 \pm 17.0	71.2 \pm 27.2	77.7 \pm 26.2	136.1 \pm 11.9
real-data-E	-3.7 \pm 1.5	9.7 \pm 5.2	11.5 \pm 6.6	15.1 \pm 8.8	60.9 \pm 13.5	10.0 \pm 7.3	14.2 \pm 8.6	61.2 \pm 14.6
Average	8.3	15.5	14.7	25.8	50.0	62.8	64.8	97.0

Table 4: **Normalized scores (mean \pm std.) for FORL and baselines with IQL.** Suffix -I denote the use of IQL algorithm [26].

maze2d-medium	IQL	IQL+LAG- \bar{s}	DMBP+LAG-I	FORL (ours)-I
real-data-A	39.5 \pm 7.8	16.0 \pm 4.9	12.2 \pm 7.7	19.5 \pm 2.9
real-data-B	3.7 \pm 7.3	13.1 \pm 8.7	14.0 \pm 9.5	32.3 \pm 13.1
real-data-C	-1.1 \pm 2.5	33.1 \pm 11.9	28.0 \pm 10.9	31.8 \pm 9.9
real-data-D	10.1 \pm 2.6	12.4 \pm 5.6	12.7 \pm 9.0	40.0 \pm 10.1
real-data-E	-4.5 \pm 0.2	12.4 \pm 5.2	9.9 \pm 2.9	24.0 \pm 8.4
Average	9.6	17.4	15.4	29.5
maze2d-large				
real-data-A	16.2 \pm 6.2	12.5 \pm 5.5	6.0 \pm 4.7	24.3 \pm 9.4
real-data-B	-0.6 \pm 2.6	3.3 \pm 7.7	13.5 \pm 10.5	42.8 \pm 9.8
real-data-C	0.1 \pm 1.5	27.8 \pm 13.1	27.9 \pm 5.5	46.7 \pm 9.2
real-data-D	1.2 \pm 3.7	11.2 \pm 7.4	8.0 \pm 7.5	23.9 \pm 8.4
real-data-E	-2.3 \pm 0.2	-1.5 \pm 1.1	1.5 \pm 4.1	7.9 \pm 7.2
Average	2.9	10.7	11.4	29.1
antmaze-umaze-diverse				
real-data-A	23.0 \pm 1.4	43.7 \pm 6.1	44.0 \pm 9.6	50.3 \pm 11.0
real-data-B	21.7 \pm 5.4	55.0 \pm 6.2	61.7 \pm 9.5	70.8 \pm 13.2
real-data-C	20.0 \pm 3.2	45.4 \pm 4.0	58.3 \pm 5.1	73.8 \pm 5.4
real-data-D	6.7 \pm 5.6	26.7 \pm 8.6	29.2 \pm 6.6	77.5 \pm 4.8
real-data-E	8.0 \pm 8.7	60.0 \pm 11.8	56.0 \pm 9.8	68.0 \pm 13.0
Average	15.9	46.1	49.8	68.1
antmaze-medium-diverse				
real-data-A	18.3 \pm 5.5	21.0 \pm 5.8	24.7 \pm 4.0	17.7 \pm 3.5
real-data-B	7.5 \pm 3.5	7.5 \pm 5.4	8.3 \pm 7.8	11.7 \pm 7.5
real-data-C	2.5 \pm 3.7	18.8 \pm 6.1	15.8 \pm 4.8	16.2 \pm 7.4
real-data-D	8.3 \pm 4.2	12.5 \pm 5.9	12.5 \pm 7.8	16.7 \pm 2.9
real-data-E	3.3 \pm 4.1	22.7 \pm 8.0	22.7 \pm 12.3	22.7 \pm 6.4
Average	8.0	16.5	16.8	17.0

F Scaling Offsets

We conduct an analysis to quantify FORL’s sensitivity to different levels of non-stationarity. We scale the offset magnitude from 0 (standard D4RL offline RL environment [15] used in training) to 1 (our experiments) using scaling factors $\{0, 0.25, 0.5, 0.75, 1.0\}$, and report performance over five random seeds across five environments in maze2d and antmaze in Figure 13. This analysis delivers two key insights: first, FORL matches baseline performance on the stationary offline RL dataset on which it was trained; second, it maintains superior results throughout the full range of non-stationarity magnitudes. Crucially, FORL’s performance degrades gracefully as the magnitude of offsets increases, whereas DQL (without forecasting) suffers steep drops. Furthermore, both

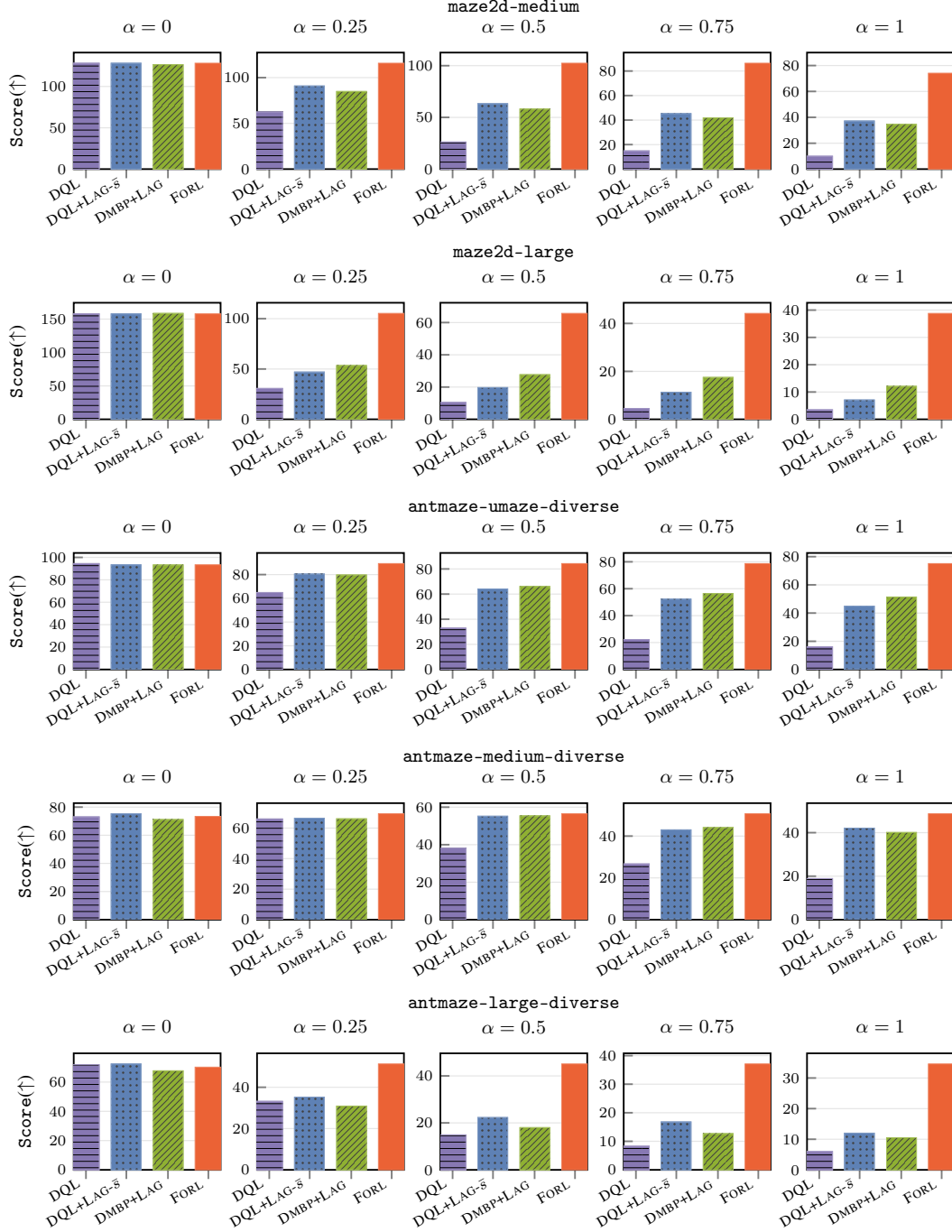


Figure 13: Average normalized scores of FORL (ours) and baselines across offset scaling factors $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$ in the maze2d-medium, maze2d-large, antmaze-umaze-diverse, antmaze-medium-diverse, antmaze-large-diverse environments. Scaling factor $\alpha = 0$ corresponds to a stationary D4RL [15] test environment; $\alpha = 1$ matches the original experimental configuration in Fig. 14. Results are averaged over 5 non-stationarities (real-data-A, real-data-B, real-data-C, real-data-D, real-data-E) and 5 random seeds.

DMBP+LAG and DQL+LAG- \bar{s} decline in a similar manner. DMBP+LAG degrades slightly more gracefully than DQL+LAG- \bar{s} in maze2d-large and antmaze-umaze-diverse.

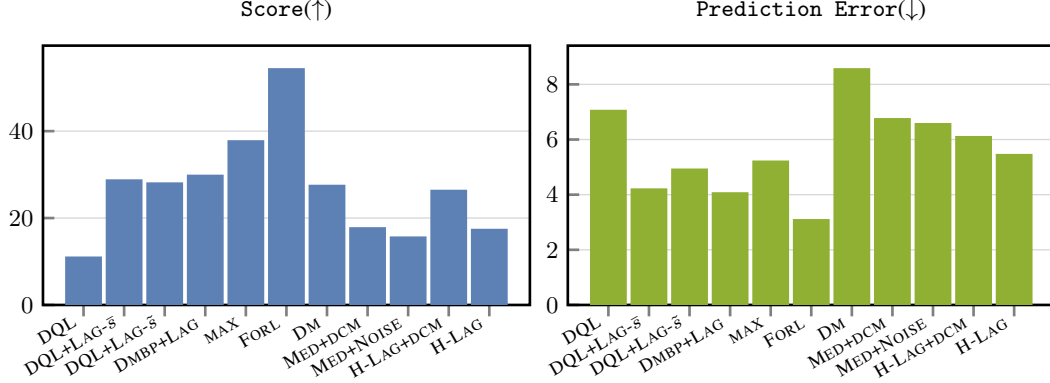


Figure 14: **Comparison of average normalized scores and prediction errors across all 25 experiments in D4RL [15]** between **FORL (ours)** and baseline methods, each evaluated over 5 random seeds. LAG denotes the integration of a zero-shot time-series foundation model [10]. While **FORL** and **MAX** also utilize this model, the subscripts are omitted for brevity.

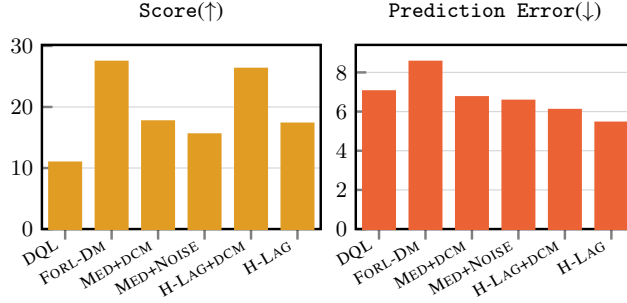


Figure 15: **Performance comparison without access to past offsets.** Average normalized scores and prediction errors for **FORL (ours)** versus baselines, aggregated over 25 experiments (5 random seeds each) in D4RL [15]. LAG denotes the integration of a zero-shot time-series foundation model [10]. However, in this setting *Zero-Shot FM* uses the samples from DM instead of past offsets.

G What if we do not have access to past offsets?

To analyze the challenging identifiability issue arising from (i) non-smoothly varying offsets and (ii) the unobservability of ground truth offsets throughout the evaluation interval, we implement a set of methods for the setting where we never have delayed access to past ground truth offsets. Fig. 14 shows the average normalized scores and prediction accuracies over 25 environment–non-stationarity pairs across five random seeds in navigation control tasks with continuous state and action spaces. Overall, these methods underperform compared to FORL. Among the cases with no access to past offsets (see Fig. 15), the best-performing methods are our proposed candidate state generation module (FORL-DM) and H-LAG+DCM, a version of FORL that utilizes *Zero-Shot FM* in addition to DM which we detailed in Section 3.1.1

FORL-DM (DM) FORL-DM, which we also refer to as DM for brevity, directly uses the state predicted by the diffusion model component. Given the multimodal nature of these candidate states, this selection does not fully leverage our framework. Yet, DM outperforms the baselines when no past offsets are used. Additional comparisons in Table 6 with other standard statistical methods like DM-MAD, DM-RANSAC, DM-RUNNING- μ , DM-RUNNING- μ -p indicate that only using the sample predicted by the diffusion model yields higher scores on average.

MED+DCM We compute the median of the predicted offsets from the previous episode, beginning with the first episode predicted by FORL’s diffusion model. We then fit a Gaussian distribution

centered at this median and sample l offsets from it, matching the sample count of *Zero-Shot FM*. Next, we apply **DCM** to these samples with the candidates generated by the diffusion model.

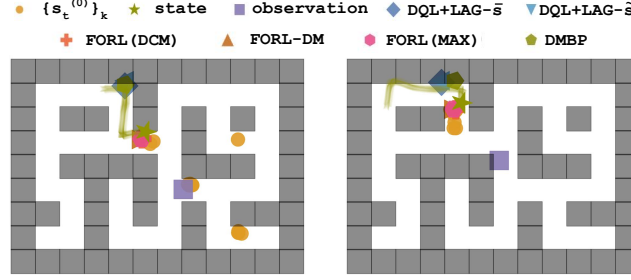


Figure 16: **Illustrations of states, observations, diffusion model predictions $\{s_t^{(0)}\}_k$, and predicted states from FORL (DCM), FORL-DM, DQL+LAG- \bar{s} , DQL+LAG- \tilde{s} , FORL (MAX), DMBP+LAG are shown.** These visualizations are from the same setting presented in Fig. 12. Maximum, minimum, and mean prediction errors across episodes for this task are provided in Table 8.

MED+NOISE We begin by computing the median offset produced by the diffusion model during the first evaluation episode similar to MED+DCM. Thereafter, we treat these offsets as evolving according to a random walk, where each offset is predicted as the previous value with white noise increments.

DM-MAD DM-MAD follows a state estimation based on robust statistics [69]. DM-MAD computes the coordinate-wise median of the differences between the observation and each denoiser prediction and discards any sample whose absolute deviation from this median exceeds $\epsilon \times$ Median Absolute Deviation (MAD). Then, it takes the median of the remaining inliers to obtain a robust offset estimate and subtracts that offset from the observation to obtain the state \hat{s}_t .

DM-RANSAC DM-RANSAC is a RANdom Sample Consensus (RANSAC) [70] based state estimation using the samples from our DM. DM-RANSAC calculates the offsets using the states generated by our DM and the observation, setting an adaptive per-dimension threshold as $\epsilon \times$ MAD. Then, it repeatedly samples a random offset candidate and chooses the candidate whose inlier set (differences within that threshold) is the largest. Then, it takes the average of those inliers to estimate the offset.

DM-RUNNING- μ DM-RUNNING- μ computes a numerically stable, global running mean [71, 72] of offsets from DM aggregated across all timesteps and episodes.

DM-RUNNING- μ -p DM-RUNNING- μ -p computes an online average of offsets [71, 72] from DM (Running- μ) per episode p . Unlike DM-RUNNING- μ , in DM-RUNNING- μ -p the statistics are reset to zero at the beginning of each episode.

H Candidate Selection

The results in Table 9, Table 8, and Table 7 show that FORL (DCM) more consistently performs better than other methods. Although the KDE-based FORL(KDE) method performs well in antmaze-medium-diverse, it significantly underperforms in cube-single-play (Table 7). Moreover, it requires bandwidth selection and a fallback mechanism to handle numerical instability, which highlights the practical advantage of DCM.

FORL (MAX), also referred to as MAX for brevity, can fail when the forecast mean of $D_{timeseries}$ is biased, misleading it to select a candidate from a geometrically distant mode that appears more likely under an inaccurate forecast. In contrast DCM, succeeds because its state estimation is not dependent on the forecast’s mean, but on a non-parametric search for the forecast sample with the highest score (minimum dimension-wise distance). Hence, DCM’s prediction error is governed by the accuracy of the forecast sample in $D_{timeseries}$ with the best score. Empirically, this yields lower maximum

Table 5: **Normalized scores (mean \pm std.) for no-access to past offsets setting.** This table shows the performance comparison of other heuristics variants using FORL-DM or leveraging a *Zero-Shot FM* in combination with FORL-DM when we do not have access to past offsets.

maze2d-medium	DQL	MED+NOISE	MED+DCM	H-LAG	H-LAG+DCM	FORL-DM
real-data-A	30.2 \pm 6.5	27.4 \pm 14.5	27.4 \pm 12.2	29.7 \pm 11.3	48.8 \pm 10.0	55.2 \pm 10.6
real-data-B	14.1 \pm 12.1	23.9 \pm 14.3	33.1 \pm 19.5	4.5 \pm 12.3	50.3 \pm 12.9	56.8 \pm 24.7
real-data-C	-2.3 \pm 3.3	17.6 \pm 8.1	-2.6 \pm 2.5	26.5 \pm 5.4	30.3 \pm 5.6	52.8 \pm 10.3
real-data-D	4.7 \pm 5.0	18.9 \pm 9.1	64.1 \pm 12.7	18.2 \pm 13.7	1.8 \pm 3.8	60.1 \pm 20.2
real-data-E	3.5 \pm 8.8	21.8 \pm 15.6	-2.1 \pm 1.6	56.7 \pm 11.6	45.7 \pm 12.4	60.5 \pm 18.2
Average	10.0	21.9	24.0	27.1	35.4	57.1
maze2d-large						
real-data-A	16.2 \pm 5.5	6.4 \pm 2.9	-1.8 \pm 0.5	7.1 \pm 3.9	7.3 \pm 2.0	11.1 \pm 4.3
real-data-B	-0.5 \pm 2.9	-0.1 \pm 1.7	-1.4 \pm 1.7	-1.4 \pm 2.3	0.9 \pm 4.5	13.4 \pm 10.3
real-data-C	0.9 \pm 1.7	3.2 \pm 4.7	-2.0 \pm 0.6	3.4 \pm 2.0	0.5 \pm 1.7	9.4 \pm 2.5
real-data-D	3.0 \pm 6.6	3.7 \pm 6.8	47.6 \pm 16.4	1.0 \pm 4.5	2.9 \pm 4.0	7.4 \pm 7.2
real-data-E	-2.1 \pm 0.4	3.8 \pm 4.7	-0.5 \pm 3.7	1.8 \pm 2.7	7.2 \pm 2.3	7.7 \pm 7.6
Average	3.5	3.4	8.4	2.4	3.8	9.8
antmaze-umaze-diverse						
real-data-A	22.7 \pm 3.0	8.3 \pm 2.4	8.3 \pm 4.7	32.3 \pm 6.3	62.7 \pm 11.2	48.7 \pm 9.1
real-data-B	24.2 \pm 3.5	9.2 \pm 5.4	25.0 \pm 13.2	41.7 \pm 5.1	33.3 \pm 2.9	56.7 \pm 8.1
real-data-C	21.7 \pm 3.5	10.8 \pm 7.7	75.8 \pm 6.4	36.7 \pm 3.2	59.2 \pm 11.5	55.0 \pm 8.7
real-data-D	5.8 \pm 2.3	17.5 \pm 11.6	6.7 \pm 6.3	10.8 \pm 2.3	42.5 \pm 7.5	54.2 \pm 7.2
real-data-E	6.0 \pm 6.8	50.0 \pm 13.5	2.0 \pm 3.0	14.0 \pm 8.3	42.7 \pm 11.9	53.3 \pm 7.8
Average	16.1	19.2	23.6	27.1	48.1	53.6
antmaze-medium-diverse						
real-data-A	31.0 \pm 6.5	34.3 \pm 10.4	55.0 \pm 6.8	15.0 \pm 4.9	17.3 \pm 6.3	4.3 \pm 0.9
real-data-B	23.3 \pm 4.8	15.8 \pm 9.9	42.5 \pm 4.6	33.3 \pm 4.2	28.3 \pm 3.5	6.7 \pm 4.8
real-data-C	10.0 \pm 2.3	23.3 \pm 5.4	10.0 \pm 4.3	31.2 \pm 4.9	40.4 \pm 6.7	4.6 \pm 1.7
real-data-D	11.7 \pm 5.4	26.7 \pm 6.3	33.3 \pm 15.9	9.2 \pm 3.5	36.7 \pm 6.8	3.3 \pm 3.5
real-data-E	18.7 \pm 4.5	26.7 \pm 18.1	0.0 \pm 0.0	14.7 \pm 5.6	46.0 \pm 16.2	6.0 \pm 4.3
Average	18.9	25.4	28.2	20.7	33.7	5.0
antmaze-large-diverse						
real-data-A	11.0 \pm 1.9	5.0 \pm 3.9	1.3 \pm 1.4	9.0 \pm 1.9	11.3 \pm 4.1	11.0 \pm 3.0
real-data-B	5.8 \pm 4.8	7.5 \pm 3.5	5.0 \pm 5.4	9.2 \pm 1.9	7.5 \pm 1.9	11.7 \pm 4.6
real-data-C	5.4 \pm 2.4	5.4 \pm 2.4	1.7 \pm 0.9	10.8 \pm 2.7	12.9 \pm 5.8	12.1 \pm 2.7
real-data-D	2.5 \pm 2.3	15.8 \pm 6.2	11.7 \pm 4.6	8.3 \pm 6.6	14.2 \pm 6.3	15.0 \pm 9.6
real-data-E	5.3 \pm 3.8	5.3 \pm 3.0	1.3 \pm 1.8	8.7 \pm 3.0	6.0 \pm 2.8	8.7 \pm 5.1
Average	6.0	7.8	4.2	9.2	10.4	11.7

Table 6: **Additional results for no-access to past offsets setting.** We present alternative ways of using FORL’s diffusion model (DM) component when we do not have access to past offsets.

maze2d-medium	DM-MAD	DM-RANSAC	DM-RUNNING- μ	DM-RUNNING- μ -p	FORL-DM
real-data-A	44.3 \pm 10.6	46.8 \pm 12.4	37.8 \pm 13.6	37.4 \pm 8.7	55.2 \pm 10.6
real-data-B	46.8 \pm 19.0	44.6 \pm 17.1	42.4 \pm 24.5	51.6 \pm 11.7	56.8 \pm 24.7
real-data-C	46.4 \pm 10.9	44.5 \pm 10.1	41.8 \pm 16.0	61.7 \pm 14.1	52.8 \pm 10.3
real-data-D	47.7 \pm 22.9	49.9 \pm 23.2	44.4 \pm 24.1	52.5 \pm 13.3	60.1 \pm 20.2
real-data-E	42.9 \pm 10.6	52.4 \pm 15.7	48.3 \pm 18.8	24.2 \pm 12.2	60.5 \pm 18.2
Average	45.6	47.6	42.9	45.5	57.1

and mean errors compared to MAX. The timeseries forecaster can generate a large set of samples that can be systematically biased, which is why we observe that the DQL+LAG- \bar{s} and DQL+LAG- \bar{s} also have high maximum prediction error in Table 8. While for this specific setting in Fig. 16, FORL (MAX) and FORL-DM are close to FORL (DCM), although worse, we observe that across the test

Table 7: Normalized scores (mean \pm std.) for FORL framework and the baselines.

cube-single-play	FQL	FORL-DM-F	FQL+LAG- \bar{s}	FORL (MAX)-F	FORL(KDE)-F	FORL-F (ours)
real-data-A	0.0 \pm 0.0	3.0 \pm 1.4	0.0 \pm 0.0	0.0 \pm 0.0	0.0 \pm 0.0	23.7 \pm 3.6
real-data-B	0.0 \pm 0.0	6.7 \pm 5.6	15.0 \pm 7.0	43.3 \pm 4.8	2.5 \pm 2.3	60.0 \pm 7.0
real-data-C	0.4 \pm 0.9	4.6 \pm 1.7	10.0 \pm 1.7	39.6 \pm 4.4	38.3 \pm 3.8	42.1 \pm 5.6
real-data-D	0.0 \pm 0.0	2.5 \pm 2.3	0.8 \pm 1.9	7.5 \pm 1.9	0.0 \pm 0.0	70.0 \pm 13.0
real-data-E	0.0 \pm 0.0	8.0 \pm 3.0	0.0 \pm 0.0	21.3 \pm 8.0	0.0 \pm 0.0	32.7 \pm 9.5
Average	0.1	5.0	5.2	22.3	8.2	45.7

Table 8: Comparison of algorithm performance on error metrics.

Algorithm	Minimum Error \downarrow	Maximum Error \downarrow	Mean Error \downarrow
FORL-DCM	0.02	2.40	0.87 \pm 0.60
FORL-MAX	0.01	9.33	2.05 \pm 1.74
DQL	2.26	11.30	5.51 \pm 2.13
FORL-DM (<i>no past offsets</i>)	0.01	9.94	3.68 \pm 2.25
DQL+LAG- \bar{s}	0.04	6.28	1.76 \pm 1.13
DQL+LAG- \tilde{s}	0.05	6.56	1.87 \pm 1.19
DMBP+LAG	0.03	6.28	1.69 \pm 1.11

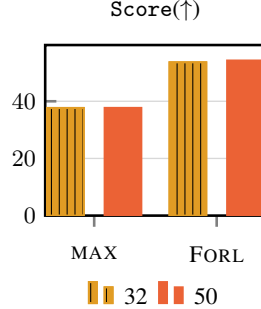


Figure 17: Average normalized scores over 25 experiments with diffusion-generated sample sizes of 32 and 50, each conducted with 5 random seeds. The results indicate that the diffusion model’s performance remains consistent across varying sample sizes, demonstrating robustness to the number of candidate states generated.

episodes, FORL (DCM) outperforms FORL (MAX) and FORL-DM in terms of maximum and mean error, demonstrating stability.

H.1 Sensitivity to Diffusion-generated Sample Size

MAX uses candidate states predicted by the diffusion model in the FORL framework and Lag-Llama but uses maximum likelihood instead of DCM. Given the multimodal nature of the candidate state distributions, we conduct a sensitivity analysis on the number of denoiser samples, a shared hyperparameter for both MAX and FORL. We report results averaged over 5 random seeds across 25 tasks (antmaze and maze2d with real-data-A, B, C, D, E). The results in Fig. 17 indicate that our diffusion model’s performance remains consistent across varying sample sizes, demonstrating robustness to the number of candidate states generated. Notably DMBP algorithm uses 50 denoiser samples.

I Zero-shot Foundation Model and Time-Series Datasets

We extract the first two univariate series from five time-series datasets: real-data-A (australian-electricity-demand) [73], real-data-B (electricity) [74], real-data-C (electricity-hourly) [73, 74], real-data-D (electricity-nips) [74, 75] and real-data-E

Table 9: **Normalized scores (mean \pm std.) for DCM candidate selection and the baselines.** Bold are the best values, and those not significantly different ($p > 0.05$, Welch’s t-test).

maze2d-medium	DM-FS- \bar{s}	DM-FS- \tilde{s}	FORL (MAX)	FORL(KDE)	FORL(DCM)
real-data-A	40.6 \pm 9.6	73.7 \pm 8.4	41.2 \pm 8.2	49.7 \pm 5.8	63.3 \pm 6.7
real-data-B	59.8 \pm 16.9	59.3 \pm 20.0	58.9 \pm 14.1	96.2 \pm 14.0	66.5 \pm 18.2
real-data-C	64.9 \pm 17.0	67.8 \pm 17.7	66.1 \pm 16.4	94.9 \pm 13.8	86.3 \pm 15.7
real-data-D	45.1 \pm 21.6	45.0 \pm 19.5	44.4 \pm 21.6	86.9 \pm 14.5	103.4 \pm 11.9
real-data-E	12.5 \pm 6.3	20.8 \pm 7.2	11.8 \pm 5.5	49.3 \pm 16.5	51.2 \pm 13.7
Average	44.6	53.3	44.5	75.4	74.1
maze2d-large					
real-data-A	11.9 \pm 5.5	20.8 \pm 6.2	11.1 \pm 2.3	6.6 \pm 2.2	42.9 \pm 4.1
real-data-B	27.9 \pm 14.7	25.1 \pm 11.7	28.3 \pm 7.1	20.9 \pm 7.2	34.9 \pm 9.2
real-data-C	34.6 \pm 6.8	32.4 \pm 5.8	34.6 \pm 13.6	36.0 \pm 7.3	45.6 \pm 4.1
real-data-D	16.4 \pm 12.0	15.5 \pm 9.0	18.4 \pm 9.9	11.8 \pm 3.2	58.4 \pm 6.5
real-data-E	8.4 \pm 5.0	7.9 \pm 3.9	9.2 \pm 6.1	5.7 \pm 5.1	12.0 \pm 9.9
Average	19.8	20.3	20.3	16.2	38.8
antmaze-umaze-diverse					
real-data-A	56.0 \pm 13.3	58.7 \pm 7.4	59.7 \pm 9.7	80.3 \pm 4.1	65.3 \pm 8.7
real-data-B	69.2 \pm 11.3	76.7 \pm 11.3	65.0 \pm 10.5	82.5 \pm 8.0	74.2 \pm 10.8
real-data-C	72.1 \pm 5.4	75.8 \pm 4.8	76.2 \pm 5.4	67.9 \pm 7.9	78.8 \pm 8.5
real-data-D	65.0 \pm 11.3	61.7 \pm 15.1	63.3 \pm 6.8	85.0 \pm 4.8	75.8 \pm 8.0
real-data-E	76.7 \pm 16.2	74.7 \pm 18.3	72.0 \pm 15.0	75.3 \pm 8.0	81.3 \pm 6.9
Average	67.8	69.5	67.2	78.2	75.1
antmaze-medium-diverse					
real-data-A	39.0 \pm 9.5	27.3 \pm 7.5	36.0 \pm 4.8	62.7 \pm 6.3	44.0 \pm 7.9
real-data-B	34.2 \pm 9.0	35.8 \pm 8.1	36.7 \pm 7.5	59.2 \pm 8.0	55.8 \pm 7.0
real-data-C	36.2 \pm 2.8	34.6 \pm 2.4	37.1 \pm 3.7	49.2 \pm 10.1	52.9 \pm 9.5
real-data-D	25.0 \pm 6.6	17.5 \pm 3.5	37.5 \pm 6.6	77.5 \pm 8.6	64.2 \pm 8.6
real-data-E	28.7 \pm 3.0	25.3 \pm 5.1	26.7 \pm 7.1	36.7 \pm 5.8	26.7 \pm 4.7
Average	32.6	28.1	34.8	57.1	48.7
antmaze-large-diverse					
real-data-A	25.0 \pm 7.9	21.0 \pm 3.5	21.7 \pm 7.9	21.7 \pm 8.3	34.3 \pm 5.7
real-data-B	25.0 \pm 5.1	30.0 \pm 7.5	20.0 \pm 6.8	47.5 \pm 15.2	46.7 \pm 11.9
real-data-C	25.8 \pm 4.6	21.7 \pm 2.4	23.8 \pm 6.4	40.0 \pm 7.6	33.8 \pm 6.8
real-data-D	14.2 \pm 7.0	15.8 \pm 5.4	21.7 \pm 7.5	35.0 \pm 14.3	46.7 \pm 12.6
real-data-E	21.3 \pm 8.4	22.0 \pm 7.7	20.7 \pm 4.9	8.0 \pm 3.0	11.3 \pm 7.3
Average	22.3	22.1	21.6	30.4	34.6

(exchange-rate², [76], [73] all accessed via GluonTS [22, 77]. Figure 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; forecasts for the remaining series and domains are provided in Figure 18.

We aim to capture a broad spectrum of scenarios for a comprehensive evaluation. For instance, the electricity-hourly dataset provides hourly electricity usage data from various consumers, while the australian-electricity-demand dataset offers 30-minute interval records of electricity demand across different Australian states. The exchange-rate dataset, on the other hand, includes daily exchange rates of multiple currencies, including those of Australia, the United Kingdom, Canada, Switzerland, China, Japan, New Zealand, and Singapore.

To effectively represent diverse offset patterns in multiple directions, we apply feature scaling to the time-series data using a normalization $x'_c = \frac{x - \bar{x}}{\max(x) - \min(x)}$ where sample mean \bar{x} , $\min(x)$ and $\max(x)$ are computed from the available data up to the context length. Furthermore, we scale these values using the minimum and maximum state values observed in the offline RL dataset [15] for navigation and minimum and maximum state values of the initial state distribution at test time for manipulation

²<https://github.com/laiguokun/multivariate-time-series-data>

Table 10: **Intra-episode non-stationarity results with $f = 50$.** We compare methods with access to past offsets (DQL+LAG- \bar{s} vs. FORL) and without (DQL vs. FORL-DM).

maze2d-medium	DQL	FORL-DM	DQL+LAG- \bar{s}	FORL
real-data-A	31.6 ± 1.6	55.7 ± 8.7	29.7 ± 9.2	17.8 ± 4.4
real-data-B	-4.7 ± 0.2	42.2 ± 9.3	63.0 ± 9.2	100.1 ± 8.5
real-data-C	-4.7 ± 0.2	39.2 ± 7.2	88.4 ± 9.4	81.0 ± 8.3
real-data-D	25.9 ± 5.3	37.0 ± 8.4	43.1 ± 8.1	101.0 ± 7.1
real-data-E	-4.7 ± 0.2	57.0 ± 10.2	5.8 ± 2.2	66.7 ± 8.3
Average	8.7	46.2	46.0	73.3
antmaze-umaze-diverse				
real-data-A	51.0 ± 7.4	47.2 ± 10.6	70.6 ± 14.5	78.2 ± 8.0
real-data-B	63.8 ± 10.2	51.2 ± 10.3	26.0 ± 16.8	61.4 ± 9.7
real-data-C	61.0 ± 9.3	53.0 ± 6.6	91.2 ± 2.5	89.0 ± 5.1
real-data-D	17.6 ± 5.9	53.8 ± 5.7	78.0 ± 4.4	90.0 ± 2.9
real-data-E	0.2 ± 0.4	51.0 ± 11.6	80.6 ± 9.7	85.6 ± 5.5
Average	38.7	51.2	69.3	80.8
antmaze-medium-diverse				
real-data-A	42.8 ± 6.9	7.2 ± 1.8	50.4 ± 2.9	54.8 ± 5.2
real-data-B	5.0 ± 3.1	37.4 ± 3.8	41.2 ± 4.8	47.2 ± 4.6
real-data-C	12.0 ± 5.1	26.8 ± 2.4	71.2 ± 2.9	61.4 ± 4.5
real-data-D	24.0 ± 4.7	30.4 ± 2.9	61.6 ± 5.0	70.2 ± 6.6
real-data-E	2.2 ± 2.3	21.0 ± 3.3	33.0 ± 5.0	37.0 ± 3.4
Average	17.2	24.6	51.5	54.1
antmaze-large-diverse				
real-data-A	13.0 ± 5.5	5.8 ± 2.6	16.0 ± 3.8	27.4 ± 3.6
real-data-B	1.2 ± 0.8	11.4 ± 4.9	19.4 ± 4.6	50.2 ± 9.7
real-data-C	4.8 ± 2.9	7.2 ± 0.8	31.8 ± 4.4	48.6 ± 4.3
real-data-D	4.6 ± 1.3	11.0 ± 2.5	25.2 ± 2.9	48.2 ± 6.2
real-data-E	2.4 ± 1.5	13.6 ± 4.8	12.8 ± 3.4	13.4 ± 2.2
Average	5.2	9.8	21.0	37.6

Table 11: **Sensitivity analysis of FORL to forecasting errors.** We compare the average prediction error (\downarrow) of our method against the DQL+LAG- \bar{s} baseline, which uses only the forecaster’s predictions. The analysis is presented across five time-series datasets. Error reduction percentages are calculated from full-precision values before rounding.

Dataset	DQL+LAG- \bar{s} (\downarrow)	FORL (\downarrow)	Error Reduction(\uparrow)
real-data-A	4.56	3.32	27.0%
real-data-B	3.66	2.29	37.4%
real-data-C	3.0	2.69	10.2%
real-data-D	4.29	1.87	56.5%
real-data-E	5.45	5.21	4.3%

[15], ensuring diverse observation spaces that can accurately represent a wide range of scenarios. We group our results in terms of time-series datasets in Table 13.

L.1 Sensitivity of FORL to Forecasting Errors

To analyze the sensitivity of FORL to forecasting errors, we compare its performance against DQL+LAG- \bar{s} , which only uses the forecaster’s predictions. Table 11 presents the average prediction error (\downarrow) across all datasets in antmaze and maze2d environments with 5 seeds.

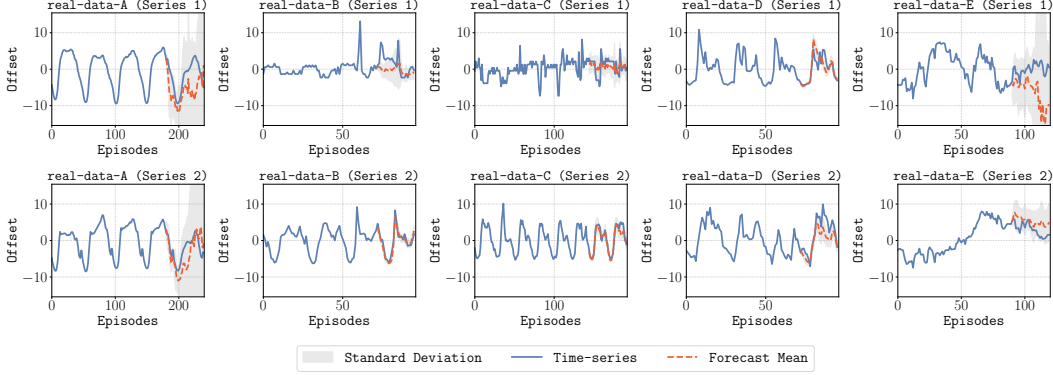


Figure 18: Zero-shot forecasting results with Lag-Llama [10] for first 2 time-series in univariate time-series datasets: `real-data-A` (Australian-electricity-demand), `real-data-E` (Exchange Rate), `real-data-B` (Electricity), `real-data-C` (Electricity Hourly), `real-data-D` (Electricity Nips).

In all datasets, FORL outperforms the DQL+LAG- \bar{s} baseline. FORL achieves its greatest impact on moderately challenging forecasts (a 56.5% error reduction on `real-data-D`). Its behavior at the extremes further demonstrates its robustness:

- FORL still refines the best forecast by 10.2% (`real-data-C`)
- FORL improves the worst forecast by 4.3% (`real-data-E`).

I.2 State Prediction Accuracy

To evaluate the state-prediction accuracy of our FORL framework, we compare it against DQL+LAG- \bar{s} . For each method, we report the mean ℓ_2 error between the true state s_t and the predicted state \tilde{s}_t obtained during evaluation for a diffusion-generated sample size of 32.

In the resulting average prediction error table in Table [12]

- Each **row** corresponds to the state estimation algorithm used at test time to generate states \tilde{s}_t , which are then provided to the policy to select actions.
- Each **column** corresponds to a method whose state estimates are evaluated on that same rollout.

The entry at row i , column j is the mean ℓ_2 error when method m_i is used in the environment, but predictions are produced by method m_j . When $i = j$, this entry measures the self-prediction error of each method; when $i \neq j$, it measures the error under an alternate method.

Across all method pairs, FORL achieves lower mean ℓ_2 errors, even in off-diagonal evaluations, demonstrating its superior state-prediction performance compared to DQL+LAG- \bar{s} . These findings are consistent with the normalized environment scores in Table [1]

J Preliminary Results for Affine Transformation with Uniform Scaling and Bias

We use the fourth series in each time-series domain to perform isotropic scaling for the dimensions affected by non-stationarity using a scaling factor of $\beta = 0.5$, with bias coming from the first two series, respectively. We apply feature scaling to time-series data with $x'_c = 1 - \beta + \beta \cdot \exp\left(\frac{x - \bar{x}}{2 \cdot (\max(x) - \min(x))}\right)$. The offset scaling for the bias uses $\alpha = 1$, which is the standard value in our experiments. As in the other ablations with scaling offsets, we use the DQL policy. Table [14] shows that FORL outperforms the baselines under this transformation. A large-scale analysis of more general transformations is left for future work.

Table 12: **Comparison of prediction errors (\downarrow).** We present state prediction accuracy for the proposed FORL framework with the baselines across 5 random seeds.

maze2d-medium	DQL+LAG- \bar{s}	FORL
DQL+LAG- \bar{s}	1.68 ± 0.46	1.35 ± 0.48
FORL	1.39 ± 0.49	1.25 ± 0.43
maze2d-large	DQL+LAG- \bar{s}	FORL
DQL+LAG- \bar{s}	2.37 ± 0.5	1.63 ± 0.57
FORL	1.91 ± 0.44	1.39 ± 0.62
antmaze-large-diverse	DQL+LAG- \bar{s}	FORL
DQL+LAG- \bar{s}	8.07 ± 1.83	5.33 ± 2.17
FORL	7.16 ± 1.71	5.89 ± 2.96
antmaze-medium-diverse	DQL+LAG- \bar{s}	FORL
DQL+LAG- \bar{s}	5.33 ± 1.15	4.75 ± 1.98
FORL	4.94 ± 1.17	4.74 ± 1.63
antmaze-umaze-diverse	DQL+LAG- \bar{s}	FORL
DQL+LAG- \bar{s}	3.51 ± 0.53	2.04 ± 0.47
FORL	3.27 ± 0.61	2.31 ± 0.51

K Offline Reinforcement Learning Environments

K.1 D4RL

We use the standard D4RL [15] offline RL environments [15] with no modifications during training, namely antmaze-medium-diverse, maze2d-medium, antmaze-large-diverse, maze2d-large, and antmaze-umaze-diverse, where initial states are randomized both in the evaluation environment and in the offline dataset. Fig. 19 illustrates the environments used from the D4RL benchmark, kitchen-complete, antmaze-large-diverse, antmaze-medium-diverse, antmaze-umaze-diverse. The maze2d-large and maze2d-medium environments share the same maze configurations as antmaze-large-diverse and antmaze-medium-diverse, respectively.

For manipulation tasks, we train on the standard kitchen-complete environment. We sample the base joint angles from $U([-1.5, 0.17])$ and the shoulder-joint angles from $U([-1.78, -1.16])$ which are set based on the minimum and maximum state space dimension intervals in the offline RL dataset to evaluate partial identifiability at test-time. The offsets affect the state dimensions associated with the base and shoulder joint angles.

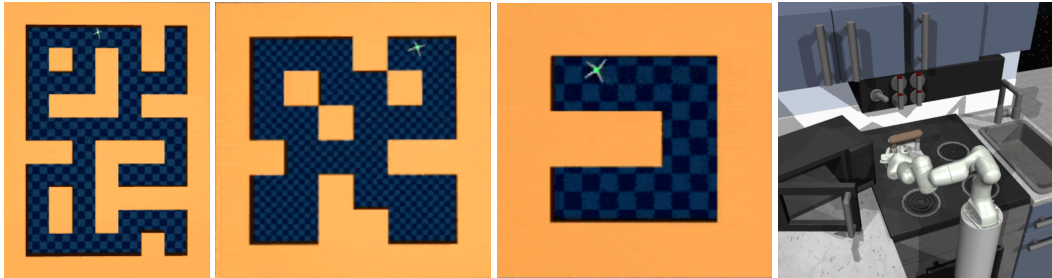


Figure 19: antmaze-large, antmaze-medium, antmaze-umaze (-v1) and kitchen-complete environments in D4RL benchmark [15].

K.2 OGBench

OGBench benchmark [21] contains both standard and goal-conditioned offline reinforcement learning tasks. To induce non-stationarity at test time, we follow the procedure from our D4RL experiments and use time-series data from GluonTS [22, 77]. For all tasks in Fig. 21, we use the default

Table 13: **Normalized scores (mean \pm std.) for FORL framework and the baselines grouped by time-series.** Bold are the best values, and those not significantly different ($p > 0.05$, Welch’s t-test).

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

Table 14: **Performance under affine observation shifts.** Normalized scores in maze2d-large with time-varying uniform scaling and bias.

maze2d-large	DQL	DQL+LAG- \bar{s}	FORL (ours)
real-data-A	5.9	6.1	39.7
real-data-B	2.4	1.6	13.8
real-data-C	2.2	22.2	32.9
real-data-D	0.7	10.7	56.1
real-data-E	-2.0	-2.3	27.5
Average	1.8	7.7	34.0

singletask-v0 variant. We report results using the FQL algorithm [23] with its officially recommended hyperparameters. For the antmaze-large-navigate environment, we use the first two time series from the GluonTS real-data-A,B,C,D,E datasets and apply an offset scaling

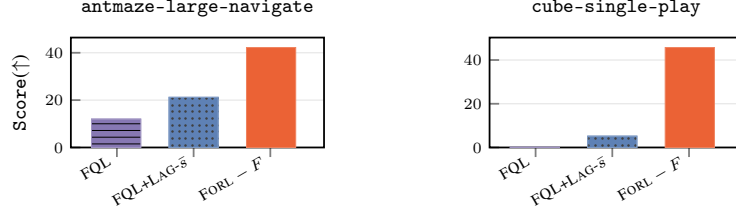


Figure 20: **Average normalized scores of FORL (ours) and baselines for OGBench**

factor of $\alpha = 0.5$. For cube-single-play, we apply offsets to the first 17 observation dimensions, which include all joint positions, joint velocities, and end effector variables (position and yaw), using the first 17 time series from each of the GluonTS real-data-A, B, C, D, E datasets. Because the real-data-A dataset only has five time series, we cycle through them repeatedly until all 17 dimensions are covered. Across all cube-single-play experiments, we use an offset scaling factor of $\alpha = 0.25$.



Figure 21: cube-single-play and antmaze-large-navigate environments in OGBench benchmark [21].

L Implementation Details

During training, we use the original offline RL dataset **without offsets**. At test-time, the offsets affect the first two state dimensions, where each offset sequence is drawn from the first two univariate time-series from diverse datasets. The agent’s policy receives only the offset-corrupted observations, with no direct access to the true underlying states throughout P episodes. The time-series forecasting model, given the past C ground-truth offsets $(b^{j-C}, \dots, b^{j-1})$, predicts the future offsets (b^j, \dots, b^{j+P}) during testing. FORL leverages these predictions and in-episode experience to dynamically adapt to unknown external perturbations. All experiments use 5 random seeds, except for (i) the preliminary affine-transformation results (Section J) and (ii) the focused error analysis across all evaluation episodes for the task shown in Fig. 16 with results reported in Table 8.

We select the hyperparameters in Table 20 based on the validation loss of the FORL diffusion model in D4RL and OGBench standard offline RL datasets. The validation loss is computed using the DM loss function in Eq. 4. Hyperparameter optimization was conducted using a grid search, with the following ranges for maze2d and antmaze: diffusion timesteps $N = \{10, 20\}$, number of hidden layers following the Temporal Unet model $\#layers = \{1, 2, 3\}$, window size $w = \{128, 256\}$, and learning rate $lr = \{0.0004, 0.0006, 0.0009\}$. For the kitchen-complete and cube-single-play $embedding\ dimension = \{64, 128\}$, learning rates $lr = \{0.0004, 0.0009\}$ and $w = \{32, 64\}$ are used for the grid search.

The architecture of our FORL Model is a noise prediction conditional TemporalUnet diffusion model [3, 39, 78]. Different from the architecture used in [3], we concatenate each element in $\tau_{(t,w)}$ with $s_t^{(n)}$ and feed it to our model without additional encoders, using the diffusion timestep embedding in the Residual Temporal Blocks. For the TemporalUnet architecture, we concatenate the Unet model output with the time-embedding before feeding it to fully connected layers, particularly in

the antmaze environments due to the large input size. The set of hyperparameters is provided in Table 20. Although the FORL conditional diffusion model is specifically utilized for time-dependent offsets in the first two dimensions of the state vector, it is trained for general-purpose state prediction, enabling it to predict all dimensions of the state in maze2d, antmaze, and OGBench environments. This approach is taken because we do not assume prior knowledge of the evaluation environment.

The method for setting seeds involves a function that initializes the seed across all relevant libraries (PyTorch, CUDA, NumPy, Gym Environment, and Python’s random module) to ensure the replicability of results. We use the open source implementation of DMBF³ with the suggested hyperparameters [56], and the pretrained Lag-Llama⁴ model [10].

M Experiments compute resources

Experiments were primarily conducted on an HPC cluster with NVIDIA A100 GPUs (40GB HBM2, PCIe 4.0/NVLink interconnect) and AMD EPYC 7302 CPUs (32 cores, 1TB RAM, 3TB local SSD), as well as on a workstation with an NVIDIA GeForce RTX 4090 (24GB GDDR6X), 128GB RAM, and a 2TB PCIe 4.0 NVMe SSD. A small portion of the experiments also ran on a cluster equipped with 4x NVIDIA V100 GPUs (16GB NVLink), 2x Intel Xeon Gold 6248R CPUs, and 384GB RAM. The total compute for published results is approximately 7,300 GPU-hours; additional failed and preliminary runs total approximately 1,500 GPU-hours.

Table 15: Hyperparameters for DQL [14, 56, 79] across kitchen-complete, maze2d, and antmaze environments.

Hyperparameters				
Maximum Timesteps	1 000 000			
γ	0.99			
τ	0.005			
Learning rate decay	true			
T	10			
β Schedule	vp			
Learning rate	3×10^{-4}			
α	0.2			
Batch Size	256			
Hidden Size	256			
Reward tune	no			
Normalize	false			
Optimizer	Adam [80]			
gn	kitchen-complete 10.0	maze2d		
		umaze-diverse:	3.0	antmaze umaze-diverse: 3.0
η	0.005	medium-diverse:	1.0	medium-diverse: 1.0
		large-diverse:	7.0	large-diverse: 7.0
		umaze-diverse:	2.0	umaze-diverse: 2.0
		medium-diverse:	3.0	medium-diverse: 3.0
MaxQ Backup	false	large-diverse:	3.5	large-diverse: 3.5
		true		true

Table 16: Hyperparameters for Implicit Q-Learning (IQL) [26, 62, 63, 81] across maze2d, and antmaze environments.

Hyperparameters	Value
Batch Size	256
Discount (γ)	0.99
Target Network Update (τ)	0.005
maze2d	$\beta = 3.0$ $\tau_{\text{IQL}} = 0.7$ Normalize Rewards = false
antmaze	$\beta = 10.0$ $\tau_{\text{IQL}} = 0.9$ Normalize Rewards = true

³<https://github.com/zhyang2226/DMBF/tree/main>

⁴<https://github.com/time-series-foundation-models/Lag-Llama>

Table 17: Hyperparameters for Flow Q-Learning (FQL) [23, 82] for cube-single-play and antmaze-large-navigate.

Hyperparameters	Value
Batch Size	256
Learning Rate	0.0003
Discount factor (γ)	0.99
Target network smoothing coefficient (τ)	0.005
BC Coefficient (α)	10.0
Flow Steps	10
Actor Hidden Dimensions	(512, 512, 512, 512)
Value Hidden Dimensions	(512, 512, 512, 512)
antmaze-large-navigate	BC Coefficient (α) = 10.0
cube-single-play	BC Coefficient (α) = 300.0

Table 18: Hyperparameters for TD3BC [56, 59, 83] across kitchen-complete, maze2d, and antmaze environments.

Hyperparameters	Value
Maximum Timesteps	1 000 000
Exploration noise	0.1
Batch Size	256
Discount factor	0.99
τ	0.005
Policy Noise	0.2
Policy Noise Clipping	0.5
Policy update frequency	2
α	2.5
Normalize	true
Optimizer	Adam[80]

Table 19: Hyperparameters for RORL [25, 56, 84] in maze2d environments.

Hyperparameters	
γ	0.99
soft_τ	0.005
Q Learning Rate	3×10^{-4}
Policy Learning Rate	3×10^{-4}
α	1.0
Auto-tune entropy	true
MaxQ Backup	false
Deterministic Backup	false
η	-1
Batch Size	256
Hidden Size	256
Target Update Interval	1
τ	0.2
Normalize	false
n sample	20
β_Q	0.0001
β_P	1.0
ϵ_{ood}	0.01
Maximum Timesteps	3 000 000
Optimizer	Adam[80]
	maze2d
β_{OOD}	0.5
ϵ_Q	0.01
ϵ_P	0.03
λ_{max}	1.0
λ_{min}	0.5
λ_{decay}	10^{-6}

Table 20: Hyperparameters for FORL across kitchen-complete, maze2d, antmaze, antmaze-large-navigate, cube-single-play environments.

Hyperparameters					
Batch Size	128				
Hidden Size	128				
# denoiser samples	50				
Optimizer	Adam [80]				
Maximum Timesteps	300 000				
	kitchen-complete	antmaze	maze2d	antmaze-large-navigate	cube-single-play
Embedding Dimension	128	64	64	64	128
w	32	256	128	256	64
Learning rate	4×10^{-4}	4×10^{-4}	9×10^{-4}	4×10^{-4}	9×10^{-4}
Observation Scale	1	1	100	1	1
Time Concatenation	true	true	false	true	true
# middle hidden layers	1	1	large: 1 medium: 3 large: 10 medium: 20	1	1
N	10	10		20	20

N Licenses for Existing Assets and Libraries

N.1 Existing Assets

- The **D4RL** [15], including the Franka Kitchen tasks, are distributed under the Creative Commons Attribution 4.0 (data) and Apache 2.0 (code) licenses as in <https://github.com/Farama-Foundation/D4RL>.
- **MuJoCo** [85] is released under the Apache 2.0 license as indicated in <https://github.com/google-deepmind/mujoco/blob/main/LICENSE>.
- **Gymnasium** (formerly OpenAI Gym) is distributed under the MIT license as indicated in <https://github.com/Farama-Foundation/Gymnasium/blob/main/LICENSE>.
- The real-data-B dataset (UCI “Electricity Load Diagrams 2011-2014”) [86] is distributed under the Creative Commons Attribution 4.0 International license as indicated in <https://archive.ics.uci.edu/ml/datasets/electricityloaddiagrams20112014>.
- The real-data-D and real-data-C variants are derived from the same UCI data and inherit the CC-BY-4.0 license.
- The real-data-A dataset [5] [73] is distributed under the Creative Commons Attribution 4.0 International license as indicated in <https://doi.org/10.5281/zenodo.4659727>.
- The real-data-E dataset introduced by Lai et al. [76] with publicly available financial data; no explicit license is provided in the original repository (<https://github.com/laiguokun/multivariate-time-series-data>), and it is used in [73], which distributes its datasets under the Creative Commons Attribution 4.0 International license.

N.2 Libraries

The libraries used in our experiments are:

1. `diffuser` uses the MIT License [6]
2. `einops` uses the MIT License [7]
3. `imageio` uses the BSD 2-Clause License [8]
4. `loguru` uses the MIT License [9]
5. `matplotlib` [87] uses a PSF-based license [10]

⁵Half-hourly demand for five Australian states

⁶<https://github.com/jannerm/diffuser/blob/master/LICENSE>

⁷<https://github.com/arogozhnikov/einops/blob/main/LICENSE>

⁸<https://github.com/imageio/imageio/blob/master/LICENSE>

⁹<https://github.com/Delgan/loguru/blob/master/LICENSE>

¹⁰<https://github.com/matplotlib/matplotlib/blob/master/LICENSE/LICENSE>

6. mujoco_py uses the MIT License.^[11]
7. numpy uses the BSD 3-Clause License.^[12]
8. pandas uses the BSD 3-Clause License.^[13]
9. scikit-video uses the BSD 3-Clause License.^[14]
10. torch (PyTorch)) is distributed under a permissive, BSD-style license that includes an express patent grant.^[15]
11. tqdm is licensed under MIT and MPL-2.0.^[16]
12. ogbench uses the MIT License.^[17]

¹¹ <https://github.com/openai/mujoco-py/blob/master/LICENSE>

¹² <https://numpy.org/doc/stable/license.html>

¹³ <https://github.com/pandas-dev/pandas/blob/main/LICENSE>

¹⁴ <https://github.com/scikit-video/scikit-video/blob/master/LICENSE.txt>

¹⁵ <https://github.com/pytorch/pytorch/blob/main/LICENSE>

¹⁶ <https://github.com/tqdm/tqdm/blob/master/LICENCE>

¹⁷ <https://github.com/seohongpark/ogbench/blob/master/LICENSE>