

Dynamics Models for Offline Hyperparameter Selection in Real-World RL

Jordan Coblin, Han Wang, Martha White, Adam White

Keywords: Applied RL, Dynamics Models, Hyperparameter Selection, Water Treatment, Industrial Control

Summary

A key obstacle to deploying reinforcement learning in real-world systems is hyperparameter selection, particularly when simulators are unavailable and online experimentation is costly. Prior work has proposed calibration models trained on offline data to approximate environment dynamics and enable offline hyperparameter selection, but these methods have so far been evaluated only in simple simulated settings. In this paper, we present the first application of calibration models in a real-world industrial setting: a municipal water treatment plant. We evaluate several calibration model approaches, including a k -nearest neighbors model with a Laplacian distance metric, on high-dimensional, non-stationary sensor data for nexting prediction tasks. Our results show that these models can generate realistic long-horizon rollouts and recover meaningful hyperparameter sensitivity trends. We further examine how calibration models scale to year-long datasets, how they support the selection of fine-tuning learning rates for pre-trained agents, and how robust they are under distribution shift. Overall, our findings provide a proof of concept for using offline dynamics models to support RL deployment in real-world environments, while highlighting important practical challenges for future work.

Contribution(s)

1. We present the first application of calibration models for offline hyperparameter selection in a real-world industrial setting, focusing on sensor prediction tasks from a municipal water treatment plant.
Context: Calibration models for offline hyperparameter selection were introduced by Wang et al. (2022), but were evaluated only in simple simulated domains. Although Janjua et al. (2023) studied prediction tasks in a water treatment plant, that work did not consider calibration models for hyperparameter selection.
2. We extend empirical evaluation methods for calibration models by introducing analyses of rollout quality, hyperparameter sensitivity, and the use of dynamic time warping to assess alignment between model-generated and true trajectories. We use these methods to compare the performance of several calibration model architectures.
Context: Previous work has focused primarily on comparing performance of the best hyperparameter configuration, but has not examined measures of model accuracy or hyperparameter sensitivity.
3. We bridge the gap toward real-world deployment by scaling calibration models to a year-long offline dataset and investigating their ability to simulate distribution shifts for the fine-tuning setting.
Context: Prior research has focused on small-scale, simulated domains and has not addressed the critical challenges of scalability and adaptation under realistic distribution shifts, both of which are essential for deploying reinforcement learning systems in practice.

Dynamics Models for Offline Hyperparameter Selection in Real-World RL

Jordan Coblin¹, Han Wang¹, Martha White^{1,2,3}, Adam White^{1,2,3}

{coblin, han8, whitem, amw8}@ualberta.ca

¹University of Alberta

²Alberta Machine Intelligence Institute (Amii)

³Canada CIFAR AI Chair at Amii

Abstract

A key obstacle to deploying reinforcement learning in real-world systems is hyperparameter selection, particularly when simulators are unavailable and online experimentation is costly. Prior work has proposed calibration models trained on offline data to approximate environment dynamics and enable offline hyperparameter selection, but these methods have so far been evaluated only in simple simulated settings. In this paper, we present the first application of calibration models in a real-world industrial setting: a municipal water treatment plant. We evaluate several calibration model approaches, including a k -nearest neighbors model with a Laplacian distance metric, on high-dimensional, non-stationary sensor data for nexting prediction tasks. Our results show that these models can generate realistic long-horizon rollouts and recover meaningful hyperparameter sensitivity trends. We further examine how calibration models scale to year-long datasets, how they support the selection of fine-tuning learning rates for pre-trained agents, and how robust they are under distribution shift. Overall, our findings provide a proof of concept for using offline dynamics models to support RL deployment in real-world environments, while highlighting important practical challenges for future work.

1 Introduction

Reinforcement learning (RL) offers adaptive, data-driven control across a range of industrial applications, including assembly line automation (Tortorelli et al., 2022), thermal power generation (Zhan et al., 2022), and commercial cooling (Luo et al., 2022). However, deploying RL in real-world systems remains constrained by numerous practical challenges. Among these, hyperparameter selection has received comparatively little attention, despite RL performance being highly sensitive to hyperparameters such as learning rate, exploration schedule, and model architecture (Henderson et al., 2018; Andrychowicz et al., 2020; Eimer et al., 2023). There are several standard approaches to hyperparameter selection in real-world tasks, each with its own limitations. Default hyperparameters (Degraeve et al., 2022) rarely reflect the dynamics of the target system and can leave substantial performance unrealized. Simulator-based tuning (Levine et al., 2016; OpenAI et al., 2019) requires an accurate simulator, which is often unavailable. Lastly, direct tuning in the real environment (Azuatalam et al., 2020; Luo et al., 2022) is often infeasible when interactions are costly, risky, or time-intensive.

A promising alternative is to use the large offline datasets commonly available in industrial systems to select hyperparameters for agents that will subsequently learn online — a setting known as Data2Online (Wang et al., 2022). This approach trains a dynamics model on offline logs and uses it as a surrogate environment, referred to as a *calibration model*. Candidate agent configurations

are evaluated through interaction with this model, and the best-performing configuration is selected for deployment and continued online learning in the real system. Unlike in standard model-based RL, the model is not used to optimize the deployment policy directly; its purpose is to preserve the relative performance of candidate hyperparameter configurations. Although prior work suggests that calibration models can approximate environment dynamics well enough for this purpose, they have so far been evaluated only in simple simulated domains, leaving their effectiveness in real-world systems unclear.

In this work, we extend the calibration model framework to sensor prediction tasks in a water treatment plant (WTP) in Drayton Valley, Alberta, Canada. Our contributions are threefold: (1) we present the first application of calibration models for offline hyperparameter selection in a real industrial system; (2) we extend evaluation methods of calibration models, incorporating rollout quality analysis, hyperparameter sensitivity, and dynamic time warping to assess trajectory similarity; and (3) we investigate scalability to large offline datasets and robustness under distribution shifts, addressing challenges critical to real-world RL deployment.

Connections to time-series forecasting and model-based RL. In the passive prediction setting studied here, calibration models resemble multivariate time-series forecasters: because the agent does not act on the plant, the model generates future sensor trajectories without conditioning on actions. This connects our setting to forecasting methods for multi-step sequence prediction (Salinas et al., 2020; Oreshkin et al., 2020; Nie et al., 2023). In control settings, by contrast, calibration models must represent action-conditioned dynamics, making them more closely related to learned dynamics models in model-based RL (Deisenroth & Rasmussen, 2011; Chua et al., 2018; Janner et al., 2019). The key distinction from both fields is the model’s purpose. Rather than minimizing forecast error or directly optimizing a deployment policy, a calibration model serves as an evaluation environment for comparing hyperparameter configurations of an agent that will subsequently learn online. Its utility therefore depends on preserving hyperparameter rankings over long rollouts, which need not coincide with predictive accuracy.

2 Background

This section introduces the framework and methods used for offline hyperparameter selection with calibration models. We first formalize the problem setting and calibration objective, then describe the k -nearest-neighbor (k NN) model used to approximate environment dynamics. We next introduce the nexting prediction task studied in our experiments and conclude with dynamic time warping, which we use to evaluate calibration-model rollouts.

2.1 Problem Formulation

Let $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$ be a dataset of N transitions sampled from a Markov decision process (MDP) under a behavior policy π_β , where $s_i \in \mathcal{S}$ is the state, $a_i \in \mathcal{A}$ is the action, $r_i \in \mathbb{R}$ is the reward, and $s'_i \in \mathcal{S}$ is the next state. A dynamics model $\hat{p}(s', r|s, a)$ is trained on $\mathcal{D}$ to approximate the environment’s true transition function $p(s', r|s, a)$.

Following Wang et al. (2022), we refer to $\hat{p}$ as a *calibration model*: it is used solely to evaluate hyperparameter configurations for an agent that will subsequently learn online.

Let $\mathcal{A}$ be a learning algorithm, Λ the hyperparameter space, and $\lambda \in \Lambda$ a hyperparameter configuration. For each λ , we define a sequence of policies $\{\pi_t^\lambda\}_{t=0}^\infty$ as the result of running $\mathcal{A}$ interactively in the environment p , i.e. $\{\pi_t^\lambda\} = \mathcal{A}(\lambda; p)$. The expected return in the true environment is defined as:

$$J_{\text{env}}(\lambda) = \mathbb{E}_p \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi_t^\lambda(\cdot|s_t), (s_{t+1}, r_t) \sim p(\cdot|s_t, a_t) \right],$$

where $\gamma \in [0, 1]$ is the discount factor. Similarly, the expected return using a calibration model is $J_{\text{model}}(\lambda) = \mathbb{E}_{\hat{p}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi_t^\lambda(\cdot | s_t), (s_{t+1}, r_t) \sim \hat{p}(\cdot | s_t, a_t) \right]$, where the policy sequence $\{\pi_t^\lambda\}$ is now learned via interaction with $\hat{p}$, $\{\pi_t^\lambda\} = \mathcal{A}(\lambda; \hat{p})$.

We define the optimal hyperparameters in each case as $\lambda_{\text{env}}^* = \arg \max_{\lambda \in \Lambda} J_{\text{env}}(\lambda)$ and $\lambda_{\text{model}}^* = \arg \max_{\lambda \in \Lambda} J_{\text{model}}(\lambda)$, and aim to learn a calibration model $\hat{p}$ such that $\lambda_{\text{model}}^* = \lambda_{\text{env}}^*$.

2.2 k NN Calibration Model

A good calibration model must be stable under long horizon rollouts, since hundreds or thousands of steps are typically necessary to evaluate a hyperparameter configuration. However, typical dynamics models are known to suffer from compounding errors, which can lead to significant divergence from the true environment over long horizons (Talvitie, 2017; Lambert et al., 2022). In order to mitigate this, Wang et al. (2022) propose using a non-parametric k NN model to estimate p by predicting next states and rewards using only transitions within the dataset, avoiding extrapolation into unobserved regions.

The k NN model samples transitions from $\mathcal{D}$ to find the k nearest neighbors of a given state-action pair according to some distance metric. Instead of using distance in the raw state-action space to find neighbors, an approximate Laplacian representation (Wu et al., 2018) can be used to construct a distance metric that is sensitive to the underlying structure of the MDP. We refer to a model that uses this distance metric as a *Laplacian k NN* model, and a model that uses distance in the raw state-action space as a *Euclidean k NN* model.

To address epistemic uncertainty, we use a *leave-one-block-out* (LOBO) ensemble of five k NN models. The dataset $\mathcal{D}$ is partitioned into five contiguous blocks, and each model is trained on a subset of $\mathcal{D}$ that excludes a different block. Results across models are aggregated by taking the *worst rank* of a given hyperparameter configuration across all models, providing a conservative estimate of hyperparameter performance.

2.3 Nexting Prediction

The *nexting prediction* problem involves an agent predicting the discounted sum of future values of an observation signal, framed as a general value function (GVF) (Modayil et al., 2012). This formulation enables temporally extended predictions about scalar signals (often called *cumulants*) in an environment. Following Janjua et al. (2023), in this work we focus on the nexting prediction problem for sensors in a WTP. For a signal o_t^i at time t , associated with channel or sensor i , the nexting value function is defined as:

$$v_t^i(s) = \mathbb{E} [G_t^i \mid s_t = s], \quad G_t^i = \sum_{k=0}^{\infty} \gamma^k o_{t+k+1}^i, \quad (1)$$

where $\gamma \in [0, 1]$ is the discount factor controlling the timescale of the prediction. This task instantiates the formulation of Section 2.1: the cumulant o^i plays the role of the reward, and the nexting target G_t^i is the corresponding return. Because the agent’s task is prediction rather than control, agent performance is measured by predictive accuracy rather than by achieved return, using the root mean squared error (RMSE) between the predicted value $\hat{v}_t^i$ and the empirical Monte Carlo return G_t^i from data trajectories across all time steps $t \in \{0, 1, \dots, T\}$. To account for differences in scale across sensors, we typically report the normalized RMSE (NRMSE), which divides the RMSE by the average return $\mu^i = \frac{1}{T+1} \sum_{t=0}^T G_t^i$. For hyperparameter selection, low prediction error thus takes the place of high return in the objective $J(\lambda)$.

2.4 Dynamic Time Warping

Dynamic time warping (DTW) is a technique for measuring similarity between time series that may differ in speed, timing, or phase (Kruskal & Liberman, 1983). Given two sequences, DTW uses warping functions ϕ_x and ϕ_y to flexibly align elements along a common time axis, allowing sections to “stretch” or “compress” for better matching, even when sequences are out of sync or unevenly sampled. Alignment is guided by constraints collectively defined by a *step pattern*; we use four common step patterns (Giorgino, 2009) to mitigate sensitivity to any single choice.

3 Real-World Application: Water Treatment Plant

To evaluate calibration models beyond simulation, we apply them to a working membrane-filtration pilot at a water treatment plant in Drayton Valley, Alberta, Canada. This setting presents high-dimensional, non-stationary, and noisy sensor data characteristic of real-world systems. Moreover, because no simulator is available and direct interaction is constrained by safety and data-collection considerations, calibration models provide a practical approach to offline evaluation.

3.1 Learning Task

The task considered throughout this section is the nexting *prediction* problem introduced in Section 2.3. At each time step, the agent predicts the discounted sum of future values for each sensor signal over a horizon determined by γ . For example, the agent may predict the near-term evolution of membrane pressure. Such anticipatory predictions provide useful knowledge about the plant, supporting system monitoring and serving as a step toward closed-loop control (Modayil et al., 2012; Janjua et al., 2023). The agent is passive: sensor signals serve as prediction targets (cumulants), there is no extrinsic reward, and the agent takes no actions that influence the plant. Although prior work has considered WTP control variables such as chemical dosing rates and backwash schedules (Liu, 2022), we focus on prediction as an initial setting in which to establish the feasibility of calibration models.

3.2 Dataset

The offline dataset consists of over two years of sensor logs (collected between 2022 and 2024) from the WTP, with 480 sensor channels sampled at 1 Hz. For our initial experiments, we use a one-week slice of the dataset, which contains $\sim 3.5 \times 10^5$ transitions. We discard sensors with consistently missing data and constant values, leaving us with a 142-dimensional feature vector for our prediction agents and calibration models. In general, we leverage the data processing pipeline outlined in Janjua et al. (2023).

For our experiments, we consider three sensors that are both critical for control of the plant and provide variety in dynamics to predict. These are the membrane pressure (PIT300), influent temperature (TIT101), and influent turbidity (TUIT101) sensors.

3.3 Experimental Setup

We train four calibration models: a LOBO Euclidean k NN ensemble, a LOBO Laplacian k NN ensemble, a feedforward neural network (NN), and a gated recurrent unit neural network (GRU). The NN and GRU architectures are trained to minimize the one-step next-state prediction loss $\mathcal{L} = \mathbb{E}_{(s_t, s_{t+1}) \sim D} \|\hat{s}_{t+1} - s_{t+1}\|^2$, and serve as parametric baselines against which to compare the non-parametric k NN models.

Evaluation terminology. Because no simulator of the WTP is available, all evaluation is based on a temporally held-out slice of sensor data. For rollout evaluation, model-generated trajectories are compared with the recorded trajectory beginning from the same initial state, which we refer to as the

true trajectory. For hyperparameter selection, the held-out sequence is replayed in temporal order to emulate the data stream encountered during passive online deployment. We refer to this replay condition as the *Online* setting and treat it as the ground-truth reference.

We evaluate each calibration model according to its ability to (1) generate realistic long-horizon rollouts and (2) support hyperparameter selection for a TD(0) prediction agent (Sutton & Barto, 2018):

- **Rollout Quality:** We generate 30k-step model rollouts and compare them qualitatively with the corresponding true trajectories. Pointwise error metrics can be difficult to interpret because otherwise similar trajectories may differ in phase, timing, or frequency. We therefore emphasize visual comparison in this initial experiment. Section 4 additionally considers dynamic time warping (DTW), while recognizing that a single scalar distance cannot fully characterize long-horizon rollout quality.
- **Hyperparameter Selection:** We sweep Adam learning rates for a TD(0) prediction agent trained from scratch using either a calibration model or the Online data stream. We report NRMSE for the NN and GRU models and worst hyperparameter rank for the ensembled k NN models. We then compare the resulting learning-rate sensitivity curves with the Online curves to assess how well each calibration model preserves hyperparameter rankings and sensitivity.

3.4 Results

In Figure 1, we compare the rollouts of the four calibration models to the true PIT300 trajectory from the held-out test data. We find that the k NN models produce sensor trajectories that resemble the true data for PIT300, with the Laplacian k NN showing closer alignment overall. For other sensors, rollouts are less accurate across all models, but the k NN models still outperform NN and GRU baselines, which tend to collapse after ~ 50 – 100 timesteps.

Figure 2 shows learning rate sensitivity curves for each model. k NN results are plotted on a separate rank axis in accordance with the ensembling strategy described in Section 2.2 — similarity in *shape* of the sensitivity curves is what we use to determine agreement. Both k NN models generally recover the correct hyperparameter rankings, with the Euclidean k NN aligning most closely with the Online curves for TIT101 and TUIT101. The weaker performance of the Laplacian k NN may stem from using a representation tuned for PIT300, highlighting the need for more robust Laplacian selection. As expected, the NN and GRU models show poor hyperparameter sensitivity, consistent with their low-quality rollouts.

4 Towards Real-World Deployment

Algorithms that perform well in controlled research experiments often face additional challenges in the complexity of real-world deployment. In this section, we take steps toward bridging this gap by exploring several modifications to the setting introduced in Section 3, broken down into three categories:

Scaling Up: In industrial settings like water treatment, years of offline sensor logs are available, presenting an opportunity to train models that capture non-stationarity, seasonality, and rare events. We extend the k NN calibration model to a full year of WTP data, amounting to ~ 32 M samples at 1 Hz, which we sub-sample by a factor of 10 to yield a more manageable ~ 3.2 M samples. The model is built in two phases—KD-tree construction and neighbor table generation—resulting in a total complexity of $\mathcal{O}((d+k)n \log n)$ (Brown, 2015), which takes approximately 10 hours on a 2 GHz Quad-Core Intel Core i5 processor for $d = 142$, $k = 3$, and $n = 3.2$ M. While computationally intensive, this process is a one-time cost, and can be further accelerated via dimensionality reduction (e.g., PCA or autoencoders), prototype selection (Wilson & Martinez, 2000), or approximate nearest-neighbor methods (Indyk & Motwani, 1998; Arya et al., 1998).

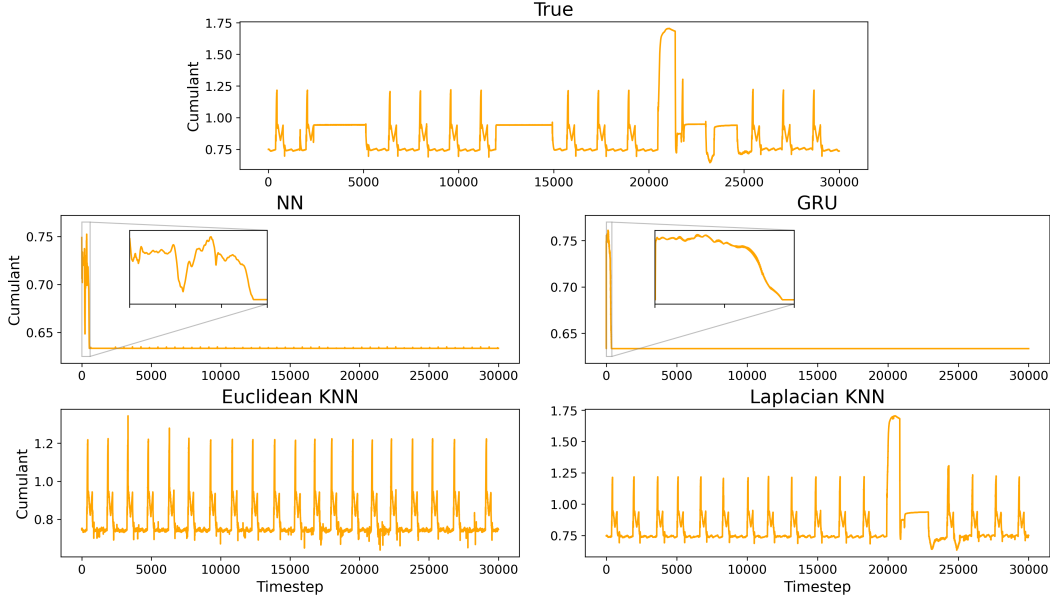


Figure 1: PIT300 sensor (membrane pressure) rollouts from the held-out test data (true) and calibration models. Each model is rolled out for 30k steps, beginning from the same start state.

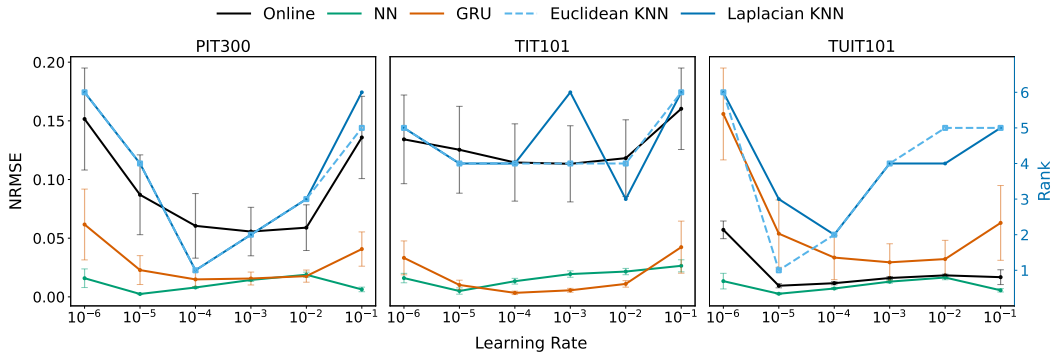


Figure 2: Learning rate sensitivity curves for a TD(0) prediction agent in the Online setting (replayed held-out data) and calibration models using the 1-week WTP dataset. Mean NRMSE and 95% confidence intervals are shown for the non-ensembled models. For the ensemble k NN models, worst rank is shown on the right y-axis. Because the two y-axes measure different quantities, curves should be compared by their shape and the relative ordering of learning rates, not by absolute values.

Agent Pre-training: In real-world settings where no simulator exists, we aim to maximize the utility of offline data. This typically involves pre-training an agent to avoid learning from scratch at deployment, after which the agent can continue to adapt online. We refer to this as the *fine-tuning* setting, and shift our focus to selecting the fine-tuning learning rate. Since the same offline data is also used to construct a calibration model, we adopt a simple partitioning strategy to separate data for pre-training and calibration, reducing overlap and better simulating a realistic transfer scenario.

Distribution Shift: System dynamics in real-world settings like the WTP can vary significantly over time due to factors such as rainfall, temperature, filter condition, and sensor drift. To evaluate how well calibration models handle such changes, we construct test sets that begin one week, one month, and three months after the training period, which correspond to April 2023, May 2023, and

July 2023 respectively. As illustrated in Figure 3, sensor patterns such as those from TIT101 change meaningfully across time, posing a challenge for generalization.

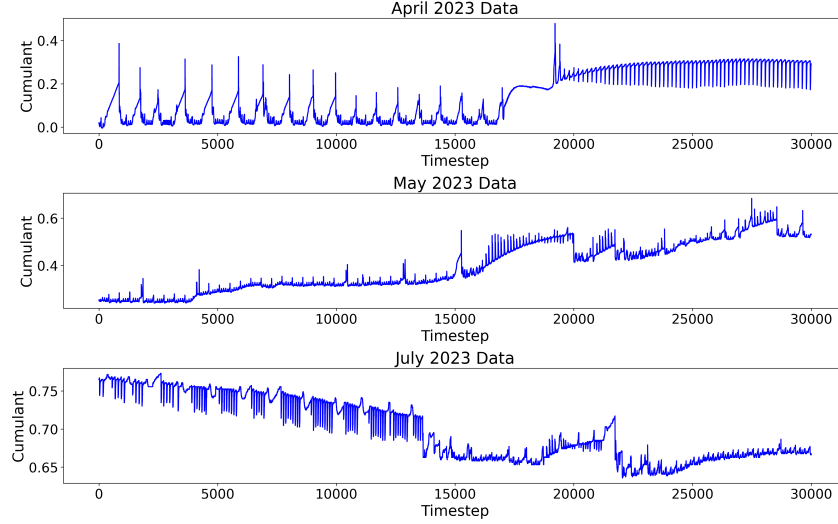


Figure 3: TIT101 sensor values across several time periods in the WTP dataset. Each plot shows roughly 3.5 days of data.

In general, it is not guaranteed that the calibration model will be able to simulate an arbitrary deployment period, since the distribution shift may be too large. However, we hypothesize that selecting a rollout start state that is representative of a deployment period can encourage the calibration model to simulate the dynamics of that period. We consider three possible strategies for selecting rollout start states $\mathcal{S}_0 = \{s_0^{[1]}, \dots, s_0^{[r]}\}$, where r is the number of rollouts we perform:

- (i) Randomly selecting $\mathcal{S}_0$ from the entire dataset (baseline method).
- (ii) Selecting $\mathcal{S}_0$ from the same calendar month in a previous year (e.g., July 2022 for a July 2023 deployment), if available.
- (iii) Selecting samples from the online deployment period and finding their nearest neighbours in the offline dataset.

In our experiments, we focus on (iii) with (i) as a baseline, as (ii) assumes seasonal consistency that may not hold due to sensor drift or evolving plant conditions. Our use of model ensembling also means that each individual start state will not exist in all ensemble models, further complicating (ii). Note that (iii) is an oracle-like strategy: it assumes access to samples from the deployment period, which would not be available when selecting hyperparameters ahead of deployment. We adopt it here as a diagnostic tool, to test whether targeted start state selection can prompt the calibration model to simulate a specific distribution shift under best-case conditions. In practice, a natural approach would be to use the most recent plant readings prior to deployment as representative start states, since these are always available and likely closest in distribution to the upcoming deployment period.

4.1 Experiments

To put the preceding ideas into practice, we evaluate our Laplacian k NN calibration model through two key questions:

Does scaling up the k NN calibration model improve generalization? We compare a k NN model trained on one year of data (*12-month k NN*), with one trained on the one-week dataset from Section 3 (*1-week k NN*). For each test period (April, May, and July 2023), we sample 30 random

states and use their nearest neighbors under the Laplacian distance metric as rollout start states, following technique (iii) from the Distribution Shift paragraph of Section 4. We evaluate rollouts both qualitatively (visual inspection) and quantitatively using DTW, where a lower DTW score indicates closer resemblance to the true sensor sequence from the test period.

Figure 4 shows a subset of PIT300 rollouts using start states from the April 2023 test set. While it is challenging to qualitatively compare rollouts, we note that the 12-month k NN is able to capture a broader range of dynamics than the 1-week k NN, with similar results for TIT101 shown in Figure 8, though not for TUIT101 (Figure 9). Quantitative results using DTW (Table 1) show mostly consistent rankings across step patterns, but mixed results across models: the 12-month k NN performs best on TIT101, the 1-week k NN on TUIT101, while results on PIT300 are mixed. Overall, these findings suggest that while the 12-month model may offer broader behavioral coverage, its generalization advantage is not conclusive. While DTW was preferable to mean squared or absolute error due to alignment issues, alternate metrics may be needed for clearer assessment (Coblin, 2024).

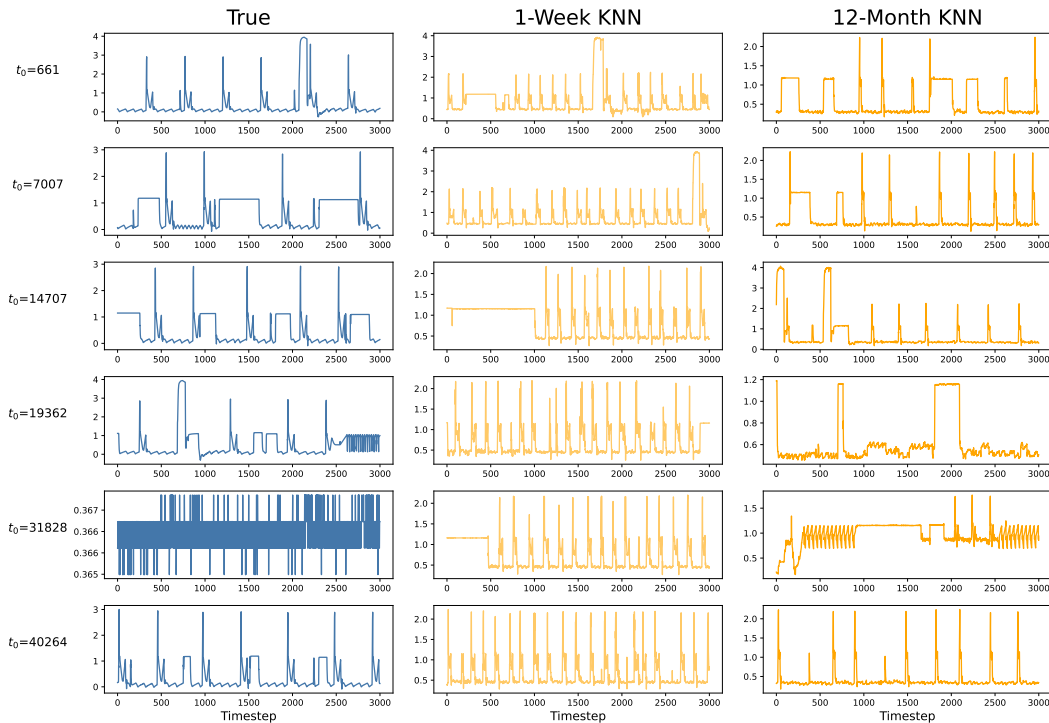


Figure 4: PIT300 rollouts for the 12-month and 1-week WTP k NN calibration models using start states t_0 from the April 2023 dataset. True rollouts from those start states are shown in the leftmost plots.

	PIT300		TIT101		TUIT101	
	1-Week	12-Month	1-Week	12-Month	1-Week	12-Month
Symmetric2	414.02 $\pm$ 0.02	483.24 $\pm$ 0.02	910.20 $\pm$ 0.05	782.05 $\pm$ 0.04	154.78 $\pm$ 0.01	194.86 $\pm$ 0.01
Asymmetric	224.0 $\pm$ 0.02	283.38 $\pm$ 0.03	523.05 $\pm$ 0.05	431.87 $\pm$ 0.04	100.92 $\pm$ 0.01	153.81 $\pm$ 0.02
SymmetricP1	805.51 $\pm$ 0.04	782.79 $\pm$ 0.04	1031.66 $\pm$ 0.05	898.61 $\pm$ 0.04	199.55 $\pm$ 0.01	268.24 $\pm$ 0.01
RabinerJuang	339.87 $\pm$ 0.03	367.87 $\pm$ 0.04	521.66 $\pm$ 0.05	447.82 $\pm$ 0.04	100.43 $\pm$ 0.01	154.09 $\pm$ 0.02

Table 1: Dynamic time warping distances computed between rollouts from k NN models (1-Week and 12-Month) and true rollouts, averaged over three test sets with 30 rollouts each for each model and sensor combination. Values are reported as mean $\pm$ 95% confidence interval. Smaller distance is better, and the best model for a specific sensor is presented in bold font. The leftmost column shows the step pattern used for the DTW algorithm.

Can the k NN calibration model simulate distribution shifts to support fine-tuning learning rate selection? To simulate a fine-tuning scenario, we pre-train a TD(0) prediction agent on the first six months of data and use the remaining six months to construct a k NN calibration model. We then use this model to guide selection of the fine-tuning learning rate for the pre-trained agent, with start states selected using technique (iii) (*Month Start States*), and using random start states as described in technique (i) (*Year Start States*) for comparison.

The Online curves in Figure 5 show that small learning rates yield good performance given a small distribution shift in the April 2023 test set. However, as the deployment period gets further away, the performance of the smallest learning rates deteriorates, indicating that the agent requires more adaptation. We find that the 6-month k NN model is able to reflect a generic distribution shift, as shown by the learning rate curves giving best performance around 1×10^{-4} to 1×10^{-5} . However, it struggles to capture *specific* shifts, as seen by its mismatch with the Online sensitivity curves. Additionally, performance between the month start states and full year start states is similar, suggesting limited benefit from our targeted start state rollout strategy. While these experiments offer a first step toward using calibration models under distribution shift, further work is needed to understand how to prompt the model to simulate dynamics from a specific deployment period, or even a period within its training data.

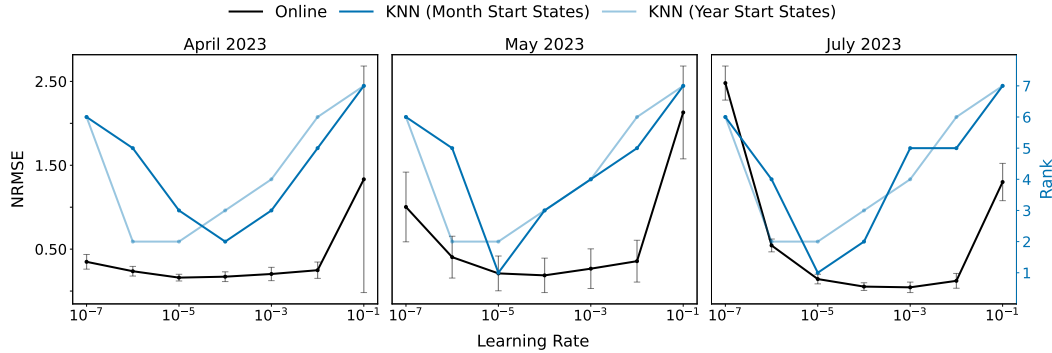


Figure 5: Fine-tuning learning rate sensitivity curves for the Online setting (replayed held-out data) and calibration models for PIT300 using six months for pre-training and six months for the calibration model. Mean NRMSE and 95% confidence intervals are shown for the Online setting. For the ensembled k NN models, worst rank is shown on the right y-axis. Because the two y-axes measure different quantities, curves should be compared by their shape and the relative ordering of learning rates, not by absolute values.

5 Conclusion

This work extends the calibration model framework for offline hyperparameter selection to a real-world industrial setting, focusing on sensor prediction tasks in a water treatment plant. We show that k NN-based calibration models can generate realistic long-horizon rollouts and preserve useful hyperparameter sensitivity. By investigating extensions involving large datasets, fine-tuning, and non-stationarity, we provide a proof of concept for leveraging offline data to guide online RL deployment.

However, important open questions remain. Under what conditions do extrapolating models fail? How should calibration models simulate distribution shifts associated with a target deployment period? To what extent is predictive accuracy necessary for preserving hyperparameter rankings? Are there more informative metrics than DTW for assessing long-horizon rollout quality? Addressing these questions will advance our understanding of calibration models and learned dynamics models more broadly, particularly in real-world systems.

References

- Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? A large-scale empirical study, 2020.
- Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.
- Donald Azuatalam, Wee-Lih Lee, Frits de Nijs, and Ariel Liebman. Reinforcement learning for whole-building HVAC control and demand response. *Energy and AI*, 2:100020, 2020. ISSN 2666-5468. DOI: <https://doi.org/10.1016/j.egyai.2020.100020>. URL <https://www.sciencedirect.com/science/article/pii/S2666546820300203>.
- Russell A. Brown. Building a balanced k-d tree in $O(kn \log n)$ time. *Journal of Computer Graphics Techniques*, 4(1):50–68, 2015. URL <https://www.jcgt.org/published/0004/01/03/paper.pdf>.
- Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.
- Jordan Coblin. Calibration models for real-world deployment of reinforcement learning agents. Master’s thesis, University of Alberta, 2024.
- Jonas Degraeve, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022. DOI: 10.1038/s41586-021-04301-9.
- Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: a model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML’11, pp. 465–472. Omnipress, 2011. ISBN 9781450306195.
- Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *Proceedings of the 40th International Conference on Machine Learning*, ICML’23, 2023.
- Toni Giorgino. Computing and visualizing dynamic time warping alignments in R: the dtw package. *Journal of Statistical Software*, 31:1–24, 2009.
- Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI’18/IAAI’18/EAAI’18. AAAI Press, 2018. ISBN 978-1-57735-800-8.
- Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613, 1998.

- Muhammad Janjua, Haseeb Shah, Martha White, Erfan Miah, Marlos Machado, and Adam White. GVF in the real world: making predictions online for water treatment. *Machine Learning*, pp. 1–31, 2023. DOI: 10.1007/s10994-023-06413-x.
- Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.
- JB Kruskal and Mark Liberman. The symmetric time-warping problem: From continuous to discrete. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, 1983.
- Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*, 2022.
- Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.
- Puer Liu. Improving water treatment using reinforcement learning. Master’s thesis, University of Alberta, Edmonton, AB, Canada, November 2022. URL <https://doi.org/10.7939/r3-7e11-kv09>.
- Jerry Luo, Cosmin Paduraru, Octavian Voicu, Yuri Chervonyi, Scott Munns, Jerry Li, Crystal Qian, Praneet Dutta, Jared Quincy Davis, Ningjia Wu, Xingwei Yang, Chu-Ming Chang, Ted Li, Rob Rose, Mingyan Fan, Hootan Nakhost, Tinglin Liu, Brian Kirkman, Frank Altamura, Lee Cline, Patrick Tonker, Joel Gouker, Dave Uden, Warren Buddy Bryan, Jason Law, Deeni Fatiha, Neil Satra, Juliet Rothenberg, Mandeep Waraich, Molly Carlin, Satish Tallapaka, Sims Witherspoon, David Parish, Peter Dolan, Chenyu Zhao, and Daniel J. Mankowitz. Controlling commercial cooling systems using reinforcement learning, 2022.
- Marlos C Machado, Marc G Bellemare, and Michael Bowling. A Laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pp. 2295–2304. PMLR, 2017.
- Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8(10), 2007.
- Joseph Modayil, Adam White, and Richard S. Sutton. Multi-timescale nexting in a reinforcement learning robot. In Tom Ziemke, Christian Balkenius, and John Hallam (eds.), *From Animals to Animals 12*, pp. 299–309. Springer Berlin Heidelberg, 2012.
- Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, 2023.
- OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation, 2019.
- Boris N. Oreshkin, Dmitri Carpo, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2020.
- David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020. ISSN 0169-2070. DOI: <https://doi.org/10.1016/j.ijforecast.2019.07.001>. URL <https://www.sciencedirect.com/science/article/pii/S0169207019301888>.

- Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Erik Talvitie. Self-correcting models for model-based reinforcement learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 2595–2601. AAAI Press, 2017. URL <https://dl.acm.org/doi/10.5555/3298483.3298613>.
- Andrea Tortorelli, Imran Muhammad, Francesco Priscoli, and Francesco Liberati. A parallel deep reinforcement learning framework for controlling industrial assembly lines. *Electronics*, 11:539, 2022. DOI: 10.3390/electronics11040539.
- Han Wang, Archit Sakhadeo, Adam White, James Bell, Vincent Liu, Xutong Zhao, Puer Liu, Tadashi Kozuno, Alona Fyshe, and Martha White. No more pesky hyperparameters: Offline hyperparameter tuning for RL, 2022.
- D Randall Wilson and Tony R Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38:257–286, 2000.
- Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.
- Xianyuan Zhan, Haoran Xu, Yue Zhang, Xiangyu Zhu, Honglei Yin, and Yu Zheng. DeepThermal: Combustion optimization for thermal power generating units using offline reinforcement learning, 2022. URL <https://arxiv.org/abs/2102.11492>.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Laplacian Distance Metric

Laplacian representations involve the graph Laplacian matrix L , which is defined as $L = D - A$, where D is the degree matrix and A is the adjacency matrix of the graph induced by the MDP, whose nodes correspond to state–action pairs. Laplacian representations have proved useful in RL for value-function approximation (Mahadevan & Maggioni, 2007), option discovery (Machado et al., 2017), and reward shaping (Wu et al., 2018), among other applications.

Let $\mathbf{u}_1, \dots, \mathbf{u}_d$ be the first d eigenvectors of the graph Laplacian L . Writing $x = (s, a)$ for a state–action pair, these define the representation

$$\psi : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}^d, \quad \psi(x) = [\mathbf{u}_1(x), \dots, \mathbf{u}_d(x)]^\top,$$

where $\mathbf{u}_i(x)$ is the value of the i -th eigenvector evaluated at state–action pair x . The distance metric is then defined as

$$d(s_i, a_i, s_j, a_j) = \|\psi(x_i) - \psi(x_j)\|_2^2, \quad x_i = (s_i, a_i), \quad x_j = (s_j, a_j).$$

Analytically computing the eigenvectors of L is typically not feasible for large graphs with an unknown transition function. Hence, we use an approximate method following the work in Wu et al. (2018) and Wang et al. (2022), which leverages spectral graph drawing to stochastically approximate the eigenfunctions of the Laplacian. Given a dataset $\mathcal{D}$, the graph drawing objective can be expressed as

$$\sum_{x_t \sim \mathcal{D}} \|\psi_\theta(x_t) - \psi_\theta(x_{t+1})\|_2^2 + \sum_{x_i, x_j \sim \mathcal{D}} \left((\psi_\theta(x_i)^T \psi_\theta(x_j))^2 - \|\psi_\theta(x_i)\|_2^2 - \|\psi_\theta(x_j)\|_2^2 \right),$$

where $\psi_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^d$ is the representation learned via a neural network with parameters θ . Intuitively, this objective is composed of an *attractive term* and a *repulsive term*. The first term is attractive insofar as it encourages ψ_θ to map state–action pairs x_t and their successors x_{t+1} closely in the representation space — this roughly captures temporal distance within an MDP. Conversely, the second term encourages independently sampled state–action pairs from the dataset to have orthogonal representations.

B Calibration Model Training Details

k NN and Laplacian Models For each k NN variant, we use a five-member LOBO ensemble with $k = 3$. To preserve sequential dependencies, the dataset is deterministically partitioned into five disjoint, contiguous blocks. Each ensemble member is trained on all data except its corresponding held-out block. The full training and calibration process proceeds in three stages:

1. (*Laplacian variant only*) Train a neural network to map raw states into a dynamics-aware Laplacian representation.
2. For each ensemble model, construct a KD-tree over its data subset in the learned representation space to support efficient nearest-neighbor queries.
3. Precompute fixed neighbor tables from the KD-trees, enabling constant-time next-state predictions during rollouts.

The underlying k NN model follows Wang et al. (2022); further implementation details for the LOBO variant in the context of the WTP are provided by Coblin (2024). Note that the LOBO ensembling method is referred to as bootstrapping in that work.

Candidate Laplacian representations are evaluated using two validation metrics. *Dynamics awareness* measures how well the representation preserves temporal relationships between states, while *representation uniqueness* detects degenerate representations in which distinct states collapse to similar embeddings. Among representations with a uniqueness score greater than 0.95, we select the one with the highest dynamics-awareness score. The corresponding hyperparameters are reported in Table 5.

NN Calibration Models The NN calibration models used a two-layer feedforward neural network architecture with a shared set of hyperparameters across sensors. Models were trained on a one-step-ahead prediction objective, mapping the current state (and optionally action) to the next state. A grid search over learning rates, hidden sizes, and batch sizes was conducted, and the best hyperparameters were selected based on performance on a validation set — see Table 3. Although multi-step prediction targets were explored, one-step predictions yielded better rollout performance in practice.

GRU Calibration Models The GRU calibration models employed a two-layer recurrent architecture designed to capture temporal dependencies in the sensor data. Models were trained on one-step-ahead prediction using sequences of historical states, with a burn-in period and sequence length tuned for each dataset. Dropout and other regularization techniques were tested to improve long-horizon stability, but simple GRU models with tuned hidden sizes and sequence lengths produced the best results. As with the NN models, hyperparameters were selected via grid search, focusing on validation performance — see Table 4.

Hyperparameter	Symbol	Water 1-Week (Online)	Water 12-Month (Pre-training)
Optimizer	-	Adam	Adam
Learning Rate	α	-	1×10^{-5}
Discount Factor	γ	0.99	0.9
Batch Size	B	256	256
Hidden Layers	-	2	2
Hidden Units	-	256	512
Replay Buffer Size	-	1M	3M
Train/Validation Split	-	-	0.9/0.1
Epochs	-	-	1000

Table 2: Hyperparameters used for the TD(0) prediction agent in WTP experiments.

Hyperparameter	Symbol	Water 1-Week
Optimizer	-	Adam
Learning Rate	α	1×10^{-3}
Batch Size	B	256
Hidden Layers	-	2
State Model Hidden Size	-	512
Epochs	-	100

Table 3: NN calibration model training hyperparameters for the WTP.

Hyperparameter	Symbol	Water 1-Week
Optimizer	-	Adam
Learning Rate	α	1×10^{-3}
Batch Size	B	256
Hidden Layers	-	2
State Model Hidden Size	-	512
Burn-in Length	-	0
Sequence Length	-	20
State Model Epochs	-	100

Table 4: GRU calibration model training hyperparameters for the WTP.

Hyperparameter	Symbol	Water 1-Week	Water 12-Month
Optimizer	-	Adam	Adam
Learning Rate	α	3×10^{-4}	1×10^{-5}
Batch Size	B	256	256
Hidden Layers	-	2	2
Hidden Units	-	256	256
Output Dimension	-	64	64
Training Steps	-	100,000	200,000
Train/Validation Split	-	0.8/0.2	0.8/0.2
Sequence Length	-	20	20
Kappa	κ	0.95	0.95
Beta	β	5	5
Zeta	ζ	0.05	0.05

Table 5: Laplacian representation training hyperparameters for the WTP.

C Experiment Details for Scaled Up Generalization Capabilities

We compare a k NN model trained on one year of data between March 31, 2022 and March 31, 2023 (*12-month kNN*), with one trained on the one-week dataset from Section 3 (*1-week kNN*). For each test period (April, May, and July 2023), we sample 30 random states and use their nearest neighbors under the Laplacian distance metric as rollout start states, following technique (iii) from the Distribution Shift paragraph of Section 4. Each rollout spans 3k steps. Hyperparameters are provided in Table 5.

D Experiment Details for Fine-Tuning Learning Rate Selection

To simulate a fine-tuning scenario, we pre-train a TD(0) prediction agent on the first six months of data and use the remaining six months to construct a k NN calibration model (i.e. an equal partitioning strategy). We then use this model to guide selection of the fine-tuning learning rate. Hyperparameters for the TD(0) prediction agent are provided in Table 2.

For each learning rate, we perform 30 runs in the Online setting and 10 per ensemble model for the calibration model, resulting in 50 total runs per α in the ensemble setting. We report NRMSE averaged over the final 25% of each run, so that early training error does not dominate the metric.

E Additional Results

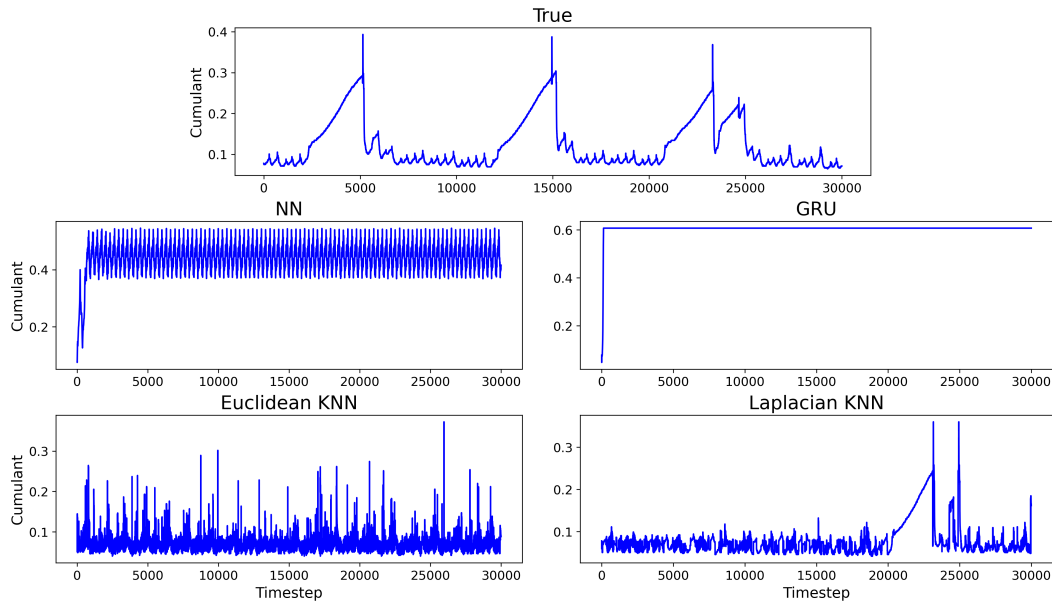


Figure 6: TIT101 sensor (influent temperature) rollouts from the held-out test data (true) and calibration models. Each model is rolled out for 30k steps, beginning from the same start state.

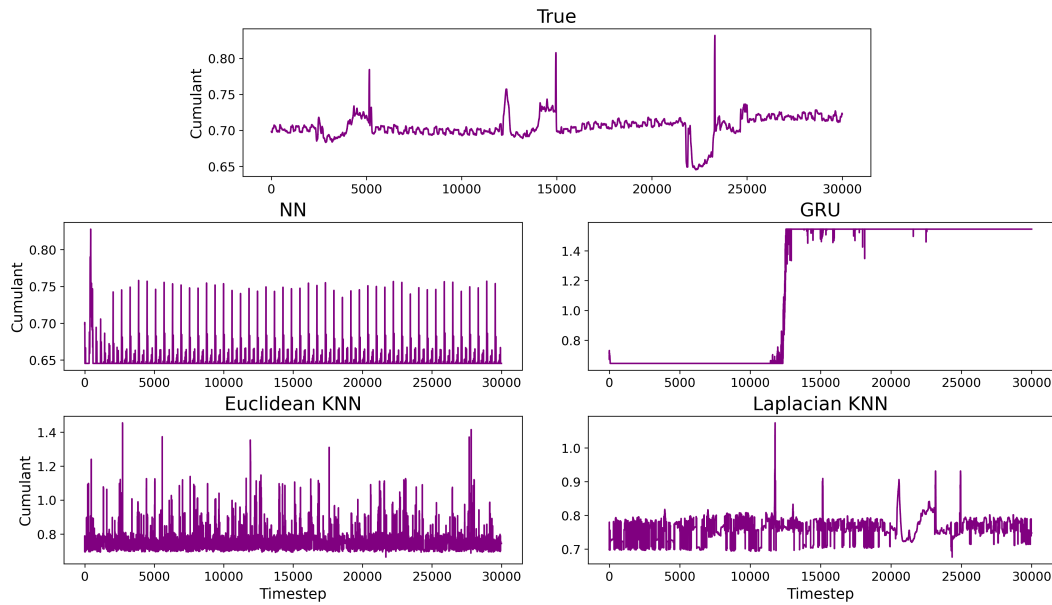


Figure 7: TUIT101 sensor (influent turbidity) rollouts from the held-out test data (true) and calibration models. Each model is rolled out for 30k steps, beginning from the same start state.

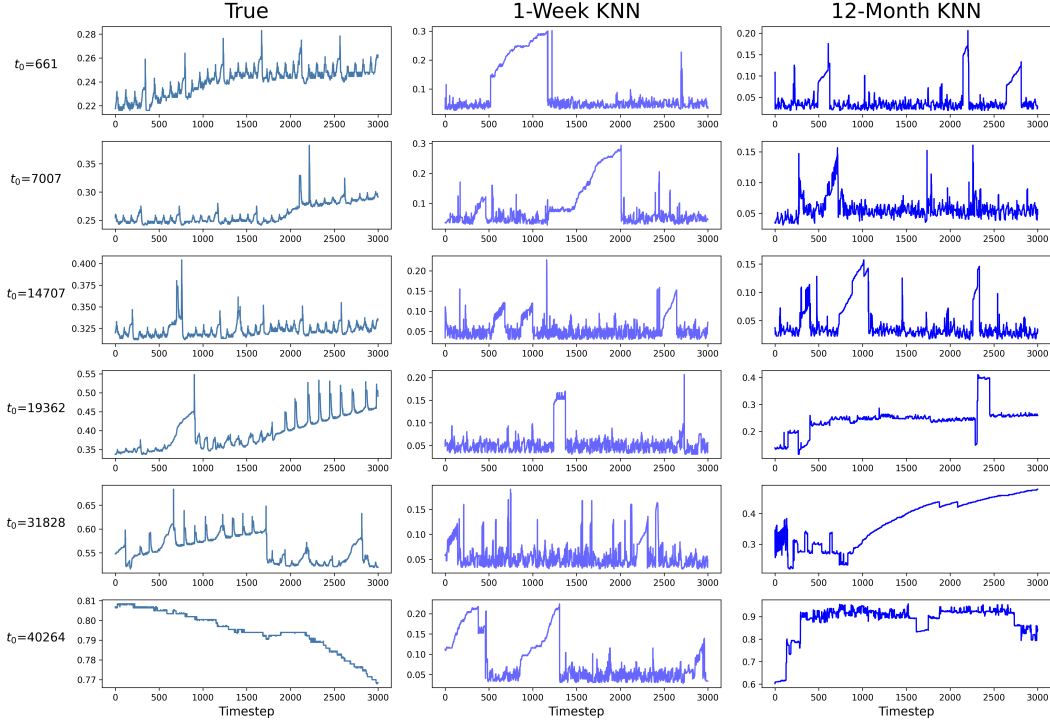


Figure 8: TIT101 rollouts for the 12-month and 1-week WTP k NN calibration models using start states t_0 from the May 2023 dataset. True rollouts from those start states are shown in the leftmost plots.

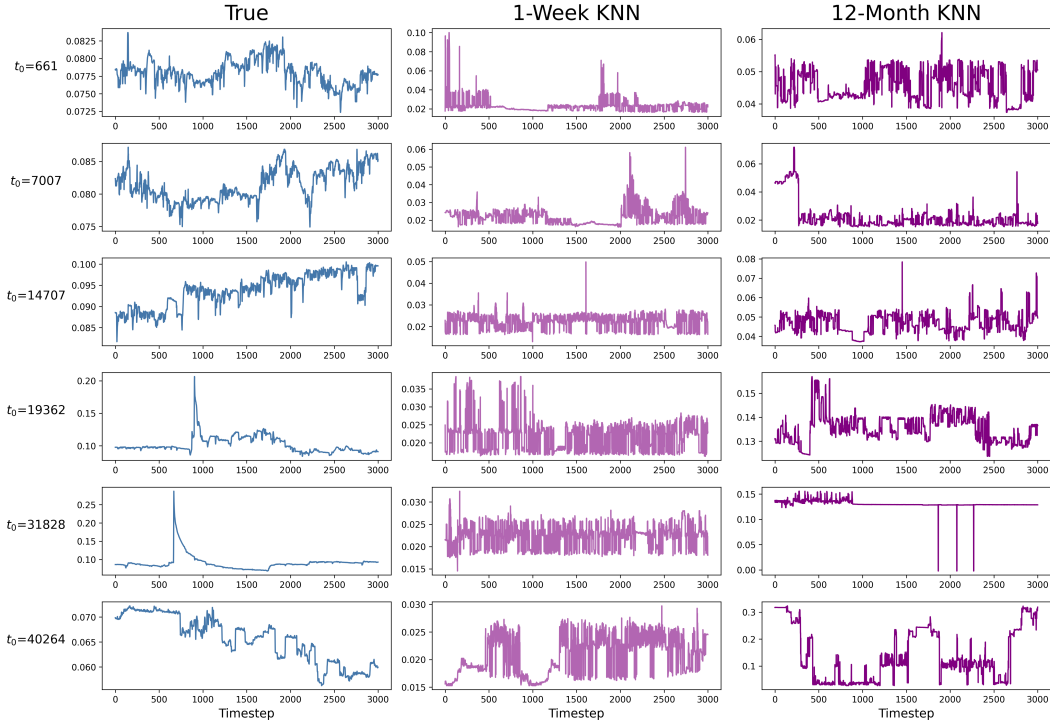


Figure 9: TUIT101 rollouts for the 12-month and 1-week WTP k NN calibration models using start states t_0 from the May 2023 dataset. True rollouts from those start states are shown in the leftmost plots.