Online Model Adaptation with Feedforward Compensation

Anonymous Author(s)
Affiliation
Address
email

Abstract: To cope with distribution shifts or non-stationarity in system dynamics, online adaptation algorithms have been introduced to update offline-learned prediction models in real time. Existing online adaptation methods focus on optimizing the prediction model by utilizing feedback from the latest prediction error. Unfortunately, this feedback-based approach is susceptible to forgetting past information. This work proposes an online adaptation method with feedforward compensation, which uses critical data samples from a memory buffer, instead of the latest samples, to optimize the prediction model. We prove that the proposed approach achieves a smaller error bound compared to previously utilized methods in slow time-varying systems. We conducted experiments on several prediction tasks, which clearly illustrate the superiority of the proposed feedforward adaptation method. Furthermore, our feedforward adaptation technique is capable of estimating an uncertainty bound for predictions.

Keywords: Online Adaptation, Optimization, Behavior prediction

1 Introduction

Robots rely on prediction models of human behavior or other agents to plan safe and efficient motions within their environment. The development or learning of accurate and reliable behavior prediction models is crucial for the successful deployment of autonomous systems in environments involving humans or diverse agents [1, 2]. However, creating a universal prediction model or algorithm capable of accurately predicting different agents in various scenarios is an extremely challenging task. Recognizing this limitation, it becomes necessary and advantageous for robots to have the capability to adapt their prediction models online when a high-fidelity behavior prediction model is not available [3, 4, 5, 6]. For instance, consider the prediction of human motion in human-robot collaboration. Due to variations in individual preferences and behavioral styles, prediction models that effectively capture one person's behavior may not be applicable to another individual. In such cases, online adaptation of the human motion prediction model becomes essential, allowing the robot to adapt its predictions by observing the specific behaviors of the individual [7, 8]. Similarly, in the context of autonomous driving, the prediction of surrounding road participants' behavior necessitates adaptation [9, 10]. Driving behavior is subject to change based on road conditions, weather, and various environmental factors. During the training phase, it is impractical to collect data encompassing all potential scenarios. Consequently, the prediction model of the agent must adapt online while observing real-time driving behaviors.

In online adaptation, a prediction model receives sequential observations, and an online optimization algorithm (such as stochastic gradient descent, SGD) is employed to update the model based on the prediction loss computed from the observed data [11]. The goal of online adaptation is to improve prediction accuracy in subsequent rounds. Most existing online adaptation approaches are based on feedback compensation [12, 13, 14], analogous to feedback control. In feedback adaptation, the prediction model exclusively incorporates the most recent data received. Following the observation

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

of a new sample, the online optimization algorithm updates the prediction model by calculating the prediction loss between the previous prediction and the latest ground truth. However, this feedback compensation approach is prone to disregarding past information, thus limiting its effectiveness.

In this work, we propose feedforward compensation in online adaptation to maximize information extraction from existing data, especially those that are more important. In the proposed **feedforward adaptation**, we will not only have forgetting as is done in conventional online adaptation [15], but also enable recalling to compensate for potential shortsighted behaviors due to forgetting. To achieve a balance between forgetting and recalling, we design a novel mechanism for feedforward compensation utilizing a memory buffer, inspired by the functionality of the Hippocampus in the human brain [16]. We maintain a memory buffer by storing recent L-steps observations (or hidden features). When similar trajectories are observed, the feedforward adaptation method will pull the corresponding data (similar sample) from the memory buffer to enhance learning. For example, in human behavior prediction tasks, a human subject may exhibit similar behavior patterns on different days. Discovering such patterns becomes challenging using conventional online adaptation that solely relies on the most recent data, whereas our feedforward adaptation method with a memory buffer facilitates their identification. The proposed feedforward adaptation is capable of estimating uncertainty bound by comparing the current observation and historical observations. Our main contributions can be summarized in the following points.

- We introduce a general online adaptation framework.
- We propose a feedforward compensation method in online adaptation. We prove that the proposed feedforward adaptation method has a smaller error bound than previously used feedback methods in slow time-varying systems.
- We propose a method for uncertainty-bound estimation.
- We conduct extensive experiments to show that the proposed feedforward adaptation is superior to conventional feedback adaptation.

2 Problem Overview

2.1 Behavior prediction

Behavior prediction is a sub-topic of time series prediction, which mainly focuses on predicting future motions (e.g. trajectory) of human or other agents given the past and current observations [17, 18, 19]. Assume the transition function of agent's behavior is denoted as f. At time step t, the input to the function is $X_t = [x_{t-I+1}, x_{t-I+2}, \cdots, x_t]$, which denotes the stack of I-step recent observations (or states). The output of the function is $Y_{t+1} = [y_{t+1}, y_{t+2}, \cdots, y_{t+O}]$, which denotes the stack of O-step future predictions. The observations x_t, y_t are vectors that may contain trajectories or features, and $x_t = y_t$ for some cases (e.g. univariate prediction).

$$Y_{t+1} = f(t, X_t). \tag{1}$$

For prediction, we use a parameterized model (e.g. Neural Networks) $\hat{f}(\theta_t, X_t)$ with learnable parameters θ_t to estimate the ground-truth transition function $f(t, X_t)$. The analysis of the online adaptation algorithms in the paper is based on the following two conditions about the local smoothness property of transition function f. Assume we maintain a L-size buffer to store recent L-steps historical observations.

Assumption 1. ***K-Lipschitz continuity condition***. *Given a time-step t, a transition function f is K Lipschitz continuious for recent L steps input data, if the following holds for $\forall s \in [t-L, t-1]$:*

$$\|f(t, X_t) - f(t, X_s)\| \leq K\|X_t - X_s\|, \tag{2}$$

where K is the bound (real number) for the change of the value of the function over input (observation) space. Intuitively, a Lipschitz continuous function is limited in how dramatically the function value can change over input space. Similar to (2), we assume our parameterized function $\hat{f}_t(\theta_t, :)$ (e.g. Neural Networks) is local Lipschitz continuous for $\forall s \in [t-L, t-1]$, with constant value $\hat{K}$:

$$\|\hat{f}(\theta_t, X_s) - \hat{f}(\theta_t, X_t)\| \leq \hat{K}\|X_t - X_s\|. \tag{3}$$

Local K-Lipschitz continuity is common for machine learning because neural networks have bounded first derivatives by proper training [20].

Assumption 2. *δ **time-varying condition.** Given a time-step t, a transition function f is δ time-varying for recent L steps under the input X_s, if the following holds for $\forall s \in [t-L, t-1]$:*

$$\|f(t, X_s) - f(s, X_s)\| \leq \delta|t-s|, \tag{4}$$

where δ is the bound (real number) for change of the value of the function over time on a fixed input observation. It is equivalent to the K-Lipschitz continuity condition over time space t, instead of input observation space X. δ time-varying condition is common for time-series tasks because many time-series have bounded time-varying (without abrupt change).

2.2 Online adaptation

Online adaptation also can be called *adaptable prediction*, since it makes an inference concurring with updating model parameters. In online adaptation, the estimate of the model parameter is updated iteratively when new data is received. Online adaptation explores local overfitting to minimize the prediction error e_{t+1}, at time step t:

$$\mathcal{L}_{err}(t) = \min_{\theta} e_{t+1} = \min_{\theta} \|Y_{t+1} - \hat{f}(\theta, X_t)\|_p, \tag{5}$$

where $Y_{t+1} = f(t, X_t) = [y_{t+1}, y_{t+2}, \cdots, y_{t+O}]$ is the ground truth observation (to be received in the future) and $\hat{Y}_{t+1} = \hat{f}(\theta_t, X_t)) = [\hat{y}_{t+1}, \hat{y}_{t+2}, \cdots, \hat{y}_{t+O}]$ is the predicted outcome from the learned model parameter θ_t. The adaptation objective can be in any ℓ_p norm.

2.3 Feedback adaptation

As discussed in Section 2.2, the goal of the online adaptation is to minimize the prediction error in the future (5). Due to the lack of ground-truth value in the current steps, it is not feasible to directly minimize the prediction error. In **feedback adaptation**, the objective of minimizing the *prediction error* in the future can be approximated by minimizing the *fitting error* in the past:

$$\mathcal{L}_{fb}(t) = \min_{\theta_t} \|Y_t - \hat{f}(\theta_t, X_{t-1})\| + \lambda \mathcal{L}_{fb}(t-1) = \min_{\theta_t} \sum_{i=1}^{t} \lambda^{t-i} \|Y_i - \hat{f}(\theta_t, X_{i-1})\|, \tag{6}$$

where $0 < \lambda \leq 1$ is a forgetting factor. At time step t, after receiving the current observations (x_t, y_t), the current samples were used to adjust the parameters of the prediction model by an online optimizer (e.g. SGD). Then a new prediction is made using the new optimized parameters. In the next time step, the estimate will be updated again given the new observation and the process repeats. It is worth noting that the observation we received at time t is y_t. The other terms in Y_t remain unknown. This paper focused on adaptation methods using one-step-ahead observation. It is also possible to conduct online adaptation with multi-step ahead observations [4].

3 Methodology

3.1 General Online Adaptation Framework

Feedback adaptation only utilizes the latest observations to adapt its models, which may result in forgetting important past information. In order to enable recalling in online adaptation, we generalize the data compensation strategy of feedback adaptation (i.e. utilizing the latest observation) to samples from recent L-steps historical observations. The general online adaptation framework is shown in Algorithm 1. At time step t, after receiving the current observations (x_t, y_t), we sample the critical input-output pairs (X_s, y_{s+1}) from recent L-steps historical observations. The critical pair will then be used to adjust the parameters of the prediction model by an online optimizer (e.g. SGD). As a special case, the critical pairs are composed by the latest observations $X_s = X_{t-1}$ in feedback adaptation.

Algorithm 1 General Online Adaptation Framework (Adaptable Prediction)

Require: Initial predictor $f(\theta_0, :)$ with parameters θ_0, Optimizer $\mathcal{O}(:,:,:)$; buffer size L
Ensure: Sequence of predictions $\{\hat{Y}_{t+1}\}_{t=1}^{T}$
1: **for** $t = 1, 2, \cdots, T$ **do**
2: Receive the ground truth observations x_t, y_t
3: Construct data for adaptation (X_s, y_{s+1}), with $t - L \leq s < t$
4: Adaptation: $\theta_t = \mathcal{O}(\theta_{t-1}, \hat{y}_{s+1}, y_{s+1})$, where $\hat{Y}_{s+1} = [\hat{y}_{s+1}, \cdots, \hat{y}_{s+O}] = \hat{f}(\theta_{t-1}, X_s)$
5: Prediction: $\hat{Y}_{t+1} = [\hat{y}_{t+1}, \cdots, \hat{y}_{t+O}] = \hat{f}(\theta_t, X_t)$
6: **end for**

3.2 Feedforward Adaptation

We propose feedforward adaptation algorithms based on the feedforward compensation, which shares similarities with feedforward control strategies [21]. The proposed **feedforward adaptation** directly minimizes prediction error (bound) by compensating for historical data without relying on feedback mechanisms. The feedforward adaptation strategy works by estimating the error bound of the prediction model and selecting historical data that can compensate for the model to minimize this error bound. Specifically, when a prediction model encounters a sample resembling very early data, the feedforward adaptation method retrieves historically similar samples to enhance the learning process. By incorporating this mechanism, the model benefits from an improved ability to make predictions without disregarding past information, thereby achieving a minimum error bound.

Feedforward adaptation aims to minimize the *upper bound of the prediction error*:

$$\mathcal{L}_{ff}(t) = \min_{\theta_{t-1}} \text{Bound}[e_{t+1}] = \min_{\theta_{t-1}} \text{Bound}[\|Y_{t+1} - \hat{f}(\theta_t, X_t)\|]. \tag{7}$$

Which is equivalent to optimizing the worst-case scenarios. The worst-case performance analysis is useful in real-world applications [22]. Specifically, minimizing the error bound of the prediction model is critical to providing robust prediction and safe planning [23]. To establish a sample selection strategy, referred to as feedforward compensation, for minimizing the error bound, we estimate the error bound as depicted in Lemma 1.

Lemma 1. ***Error Bound of Online Adaptation***. *Under the K-Lipschitz continuity condition and δ time-varying condition (2) to (4), the (prior) prediction error e_{t+1} of general online adaptation (Algorithm 1) has the following upper bound:*

$$e_{t+1} \leq K\|X_t - X_s\| + \delta|t - s| + \|Y_{s+1} - \hat{f}(\theta_t, X_s)\| + \|\hat{f}(\theta_t, X_s) - \hat{f}(\theta_t, X_t)\| \tag{8}$$

$$\leq (K + \hat{K})\|X_t - X_s\| + \delta|t - s| + \|Y_{s+1} - \hat{f}(\theta_t, X_s)\| \tag{9}$$

The error bound estimation comprises three components. The first term accounts for the difference between the current sample X_t and the selected critical sample X_s; the second term considers the time difference between current step t and the step s associated with the selected critical samples; the third term represents the (posterior) fitting error of input-output tuple (X_s, Y_{s+1}), which tends to be small through appropriate fitting techniques. The proof can be found in Appendix A.

Directly minimizing the error bound (9) is impractical due to the unavailability of information regarding the Lipschitz constant K and time-varying factor δ. However, in a slow time-varying system, it is possible to simplify the aforementioned error bound.

Slow time-varying system is one whose transition function (or behavior) changes slowly over time, i.e. $\delta \approx 0$. The slow time-varying system only assumes the transition function f varying slowly (locally) within recent L-steps not (globally) for every step. Slow time-varying systems are common in real-world applications because we can choose the adaptation step to be small or the adaptation rate to be fast compared to the changing rate of the dynamics.[1] We describe the detailed condition

[1] In some cases, we can convert the non-slow-varying system into a slow-varying system by differencing, more details are shown in Section 4.2.

for feedforward adaptation in Section 4.1. In the slow time-varying system, let $\delta \approx 0$ in the error bound (9), the optimization objective for the error bound can be simplified as below:

$$\min_{\theta_t} \text{Bound}[e_{t+1}] = \min_{\theta_t, s}(K + \hat{K})\|X_t - X_s\| + \|Y_{s+1} - \hat{f}(\theta_t, X_s)\| \tag{10}$$

The unknown parameter K still makes it difficult to directly optimize the above objective. In order to make the objective feasible, we change the joint minimization over s and θ_t to a bi-level optimization which first minimizes the first term of objective (10) over the sampling time step s, then minimizes the second term of the objective over parameter θ_t. Thus, the simplified objective function becomes:

$$\mathcal{L}_{final} = \min_{\theta_t} \|Y_{s^\star+1} - \hat{f}(\theta_t, X_{s^\star})\| \tag{11}$$

$$\text{s.t.} \quad s^\star = \arg \min_{i \in [t-L, t-1]} \|X_t - X_i\| \tag{12}$$

In summary, the proposed feedforward adaptation method selects the most similar samples to the current observation as the critical pair $(X_{s^\star}, Y_{s^\star+1})$ by (12). Then using the critical pair to optimize the prediction model by (11).

3.3 Uncertainty Estimation

The error bound (8) provides uncertainty estimation of the prediction results. Here we use estimation of $\tilde{K}_t$ and $\tilde{\delta}$ to approximate real K and δ in (8). We use confidence factor $\sigma \in (0, 1]$ to decay the error bound . The uncertainty estimation $\hat{U}_{t+1}$ for prediction $\hat{Y}_{t+1}$ is shown below:

$$\hat{U}_{t+1} = \sigma \cdot (\tilde{K}_t\|X_t - X_s\| + \tilde{\delta}(t-s) + \|Y_{s+1} - \hat{f}(\theta_t, X_s)\| + \|\hat{f}(\theta_t, X_s) - \hat{f}(\theta_t, X_t)\|) \tag{13}$$

The confidence factor σ is a predefined hyperparameter, e.g. $\sigma = 0.9$ for 90% confidence of uncertainty estimation. $\tilde{\delta}$ and $\tilde{K}_t$ are also predefined hyperparameters, e.g. $\tilde{\delta} = 10^{-4}, \tilde{K} = 1$. FIn order to enhance the accuracy of the uncertainty estimation, we have the option to iteratively update $\tilde{K}t$ based on the estimated uncertainty $\hat{U}t$ and the actual error e_t. Such that if the previous uncertainty estimation is much larger than the real error then we shrink the $\tilde{K}_t$ value. The $\tilde{K}_t$ estimation criteria are shown below,

$$\tilde{K}_t = \tilde{K}_{t-1} + \beta \cdot \frac{e_t - \hat{U}_t}{\|X_t - X_s\|}, \quad \beta \in [0, 1] \tag{14}$$

The overall feedforward adaptation algorithm is shown in Algorithm 2.

Algorithm 2 Online Adaptation with Feedforward Compensation

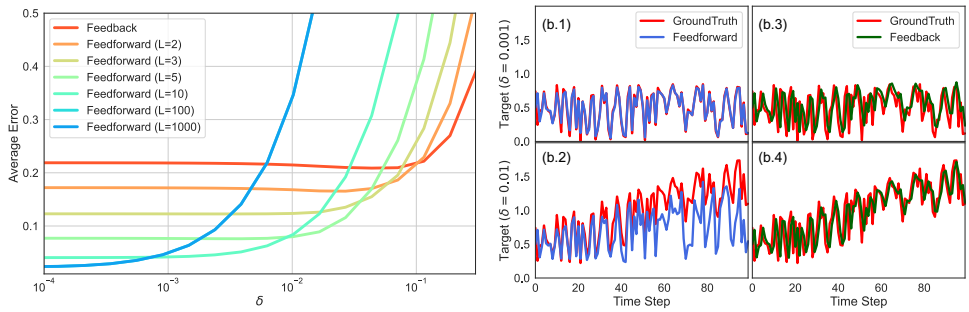
Require: Initial predictor $f(\theta_0, :)$ with parameters θ_0, Optimizer $\mathcal{O}(:, :, :)$, L-size buffer B
Ensure: Sequence of predictions $\{\hat{Y}_{t+1}\}_{t=1}^T$ and estimated uncertainty $\{\hat{U}_{t+1}\}_{t=1}^T$

```
for t = 1, 2, ···, T do
    Receive the ground truth observation values x_t, y_t; Construct input X_t = [x_{t-I+1}, ···, x_t]
    Find the critical (similar) input-output pairs (X_{s*}, y_{s*+1}) from buffer B by (12)
    Adaptation by (11): θ_t = O(θ_{t-1}, ŷ_{s*+1}, y_{s*+1})
    Prediction: Ŷ_{t+1} = [ŷ_{t+1}, ···, ŷ_{t+O}] = f(θ_t, X_t)
    Uncertainty Û_{t+1} Estimation by (13) and (14)
    Add current data to buffer: B.append(X_t, y_t)
    if size(B) > L then
        B ← keep_more_recent_samples (B, L)
    end if
end for
```

4 Analysis

4.1 Comparison between Feedforward and Feedback Adaptation

The main difference between feedforward adaptation and feedback adaptation lies in the critical sample selection strategy that used in the optimization. In feedback adaptation, the critical sample



(a) Average error (b) Prediction output

Figure 1: (a) Average error of feedback and feedforward adaptation with different buffer sizes L and time-varying factors δ. (b) Prediction output of feedforward (blue curve) and feedback adaptation (green curve) with buffer size $L = 100$ and time-varying factor $\delta = 10^{-3}$ and $\delta = 10^{-2}$.

is composed by the latest observation $X_s = X_{t-1}$. In the proposed feedforward adaptation, the critical sample is the most similar sample to the current observation within recent L-steps $X_s = \arg\min_{X_i} \|X_t - X_i\|$. In the following analysis, we assume there is no fitting error and mainly focus on the data selection strategy. The important findings regarding the expected error bound for both feedforward and feedback adaptation are presented in Lemma 2. Proof of these results can be found in Appendix B.

Lemma 2. ***Expected Error Bound***. *Considering a transition function $f(t, X)$ and a parameterized model $\hat{f}(\theta, X)$ subject to the K-Lipschitz continuity condition and δ time-varying condition (2) to (4). Let $E[|X_t - X_{t-1}|] := D$ represent the expected distance between consecutive samples, and $E[\min_{X_i} |X_t - X_i|] := D^\star$ denote the expected minimum distance. We have the following results for error bound for feedback adaptation B_e^{fb} and error bound for feedforward adaptation B_e^{ff}.*

(a) Expectated error bound for feedback adaptation: $E[B_e^{fb}] = (K + \hat{K})D + \delta$.

(b) Expected error bound for feedforward adaptation: $E[B_e^{ff}] \leq (K + \hat{K})D^\star + \delta L$.

(c) If time-varying factor δ is smaller, specifically $\frac{\delta}{K+\hat{K}} < \frac{D-D^\star}{L-1}$, feedforward adaptation has a smaller expected error bound than feedback adaptation.

(d) If the inputs are random variables sampled from the uniform distribution, $X_t \sim \mathcal{U}(0, 1)$, we have more concrete conclusions. (1) $E[B_e^{fb}] = \frac{K+\hat{K}}{3} + \delta$; (2) $E[B_e^{ff}] = \frac{K+\hat{K}}{L+2} + \delta L$; (3) If $\frac{\delta}{K+\hat{K}} < \frac{1}{12}$, feedforward adaptation with buffer size $L = 2$ outperforms feedback adaptation. The optimal buffer size for minimum error bound is $L^\star = \sqrt{\frac{K+\hat{K}}{\delta}} - 2$.

Numerical Example. We design a toy example to evaluate Lemma 2, where we use a neural network to learn the following linear time-varying system.

$$y_{t+1} = f(x_t) = \sin x_t + \alpha t, \quad x_t \sim \mathcal{U}(0, 1)$$

The results in Fig. 1a validate the theoretical results from Lemma 2(d). (1) As shown in the figure, for a smaller time-varying factor δ, feedforward adaptation with a larger buffer size L achieves a smaller prediction error. However, if δ is larger, the performance of the feedforward adaptation deteriorates. (2) If $\delta < 0.1$, feedforward adaptation with buffer size $L = 2$ outperforms feedback adaptation. Conversely, if $\delta > 0.1$, feedback adaptation is better. The threshold $\delta^\star = \frac{1}{12}(K + \hat{K}) \approx 0.1$ aligns wit Lemma 2(d). Further discussion can be found in Appendix B.3.

Figure 1b illustrate the comparison of prediction results between feedforward and feedback adaptation. The first row shows the prediction results for $\delta = 10^{-3}$ using feedforward adaptation with $L = 100$ (b.1) and feedback adaptation (b.3). As shown in the figure, feedforward adaptation results in better prediction. The second row shows the prediction results for $\delta = 10^{-2}$. Compared to feedback adaptation, feedforward adaptation places more emphasis on the observation-dependent aspect of the ground truth function and gives less consideration to the time-varying shift of the function.

4.2 Applications of Feedforward Adaptation

According to Lemma 2(c), feedforward adaptation outperforms feedback adaptation when $\delta \approx 0$, regardless of the values of $\delta, K, D, D^\star$. To aid in making this determination, we propose a straightforward criterion based on time series stationarity analysis.

A stationary time series is one whose properties do not depend on the time at which the series is observed [24]. Fora time series (X_t, Y_{t+1}), stationarity implies that the transition function $f : X_t \to Y_{t+1}$ has a time-varying factor δ close to zero. To determine the stationarity of a given time series signal (e.g., the training set of a prediction task), we employ the ADF test [25]. If the time series is found to be stationary, indicating $\delta \approx 0$, feedforward adaptation can be used to achieve improved results. If the original time series is non-stationary, we can employ differencing to transform the series and assess stationarity again based on the differenced signal. Differencing calculates the difference between two consecutive observations [26], i.e. predicting $d_{t+1} = Y_{t+1} - Y_t$ instead of Y_{t+1}. It stabilizes the mean of a time series and thus reduces the trend. Differencing has the ability to convert many non-stationary series into stationary ones. For a more explanation, please refer to Appendix C. The criterion and procedure for applying feedforward adaptation are shown in Fig. 2.

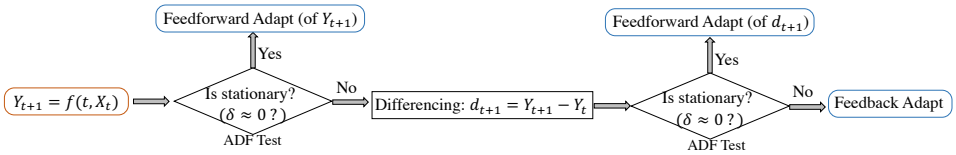


Figure 2: The criterion and procedure for applying feedforward adaptation.

5 Experiments

In this section, we evaluate the proposed feedforward adaptation on three scenarios, (1) Human motion prediction in human-robot collaboration: We use the THOR dataset [27] for human motion prediction and the Assembly dataset [28] for arm motion prediction; (2) Vehicle trajectory prediction in autonomous driving: The NGSIM dataset [29] is employed for evaluating vehicle trajectory prediction; (3) Robotic arm trajectory prediction for quality control and monitoring purposes: We evaluate the method on robotic arm trajectory prediction tasks in pick-and-place scenarios. More details of the dataset can be found in Appendix D.1.

We performed the ADF test on the datasets to examine their slow-time-varying properties, as shown in Appendix D.2. The results indicate that the transition function for the THOR and Assembly datasets exhibits slow-time-varying. On the other hand, the transition function for the NGSIM and Robot arm datasets does not demonstrate slow-time-varying behavior. However, their corresponding difference series meet the slow-time-varying condition. Therefore, we apply feedforward adaptation to the trajectory data of the THOR and Assembly datasets, and to the velocity (difference series) data of the NGSIM and Robot arm datasets.

Experimental Design. We employ a Multi-layer Perceptron (MLP) with direct multistep (DMS) prediction strategy [30] as our parameterized prediction model $\hat{f}(\theta, :)$. The DMS strategy has shown superior performance compared to larger Transformer-based models, as reported in [30]. We first train prediction models on the train set. In the evaluation phase, we simulate real-world applications by incrementally receiving observations from the test set. At each time step, we perform online adaptation by optimizing the model using stochastic gradient descent (SGD) with selected previous observations, employing either feedback or feedforward compensation. Subsequently, we obtain the prediction output from the updated model. The prediction results are evaluated using the root-mean-squared error (RMSE) metric.

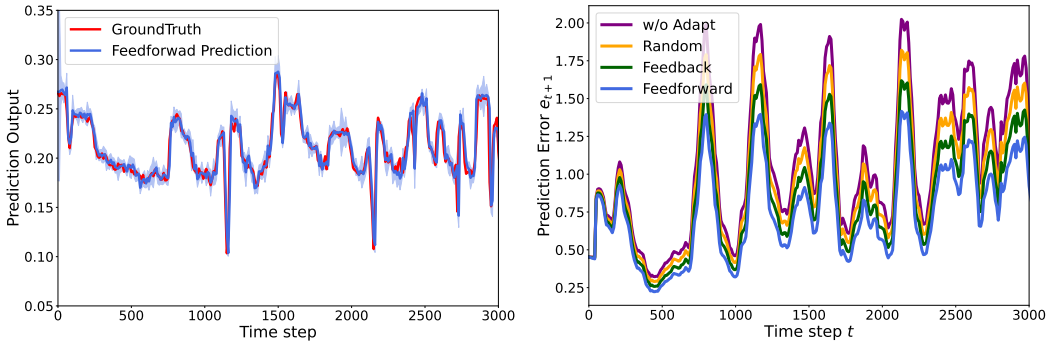
Baselines. We compare the proposed method with four baselines. 1) *w/o adapt* performs prediction without any adaptation, serving as a lower bound for adaptation methods. 2) *Feedback adaptation* is the most important baseline to us. 3) *Random adaptation* selects the critical pair from the buffer

using random sampling. 4) *Full adaptation* utilizes all samples from the buffer to adapt the model, resembling offline training.

Results. The detailed results are shown in Table 1. Feedforward adaptation demonstrates superior performance across all four datasets. On average, feedforward adaptation outperforms feedback adaptation by 7.5% in terms of RMSE. Moreover, feedforward adaptation surpasses full adaptation, indicating that adapting models with more samples is not effective. Online adaptation focuses on the ability to rapidly learn and adapt in the presence of non-stationarity, rather than emphasizing generalization ability as seen in offline learning. Furthermore, feedforward adaptation outperforms both feedback adaptation and random adaptation, highlighting the significance of the sample selection strategy based on sample similarity.

Table 1: Performance (RMSE) comparison between the proposed feedforward adaptation method and other baselines. We use **boldface** and underline for the best and second-best results.

Data\Method	w/o Adapt	Random Adapt	Full Adapt	Feedback Adapt	Feedforward Adapt
THOR (m)	1.208	1.052	0.994	0.891	**0.836**
Assembly (cm)	1.324	1.229	1.210	1.196	**1.171**
NGSIM (m)	1.203	1.113	1.007	0.995	**0.890**
Robot arm (rad)	0.257	0.229	0.228	0.214	**0.193**



(a) Prediction and uncertainty

(b) Online prediction error

Figure 3: Experimental results on Assembly dataset. (a) A segment of the prediction output and uncertainty estimation of the proposed feedforward adaptation. For simplicity, only the prediction for the future 10th step, one dimension is plotted. The blue dashed region represents the uncertainty. (b) Comparison of prediction error between the feedforward adaptation and other baselines.

One advantage of the proposed feedforward adaptation is its capability to provide uncertainty estimation, as shown in Algorithm 2. Figure 3a shows the prediction output (blue curve), ground truth label (red curve), and uncertainty estimation (blue dashed region) on the Assembly dataset. It is evident that the estimated uncertainty generally encompasses the actual ground truth value, confirming the effectiveness of the proposed uncertainty estimation. Figure 3ashowcases the real prediction error for different adaptation methods over time, with feedforward adaptation exhibiting the smallest error. Furthermore, when dealing with periodic data, feedforward compensation has the inherent ability to capture the underlying periodicity in the time series, as demonstrated in Appendix D.5.

6 Conclusion, Limitation, and Future work

This paper investigates an effective feedforward adaptation algorithm for behavior prediction tasks. We provide evidence that feedforward adaptation exhibits a smaller error bound compared to conventional feedback adaptation in slow time-varying systems.

One of the limitations of the work is that the proposed feedforward adaptation method only works well on slow-varying systems. In the future, we plan to explore the integration of feedforward and feedback adaptation in more general systems, aiming for a more comprehensive and versatile approach.

References

[1] Y. Cheng, L. Sun, C. Liu, and M. Tomizuka. Towards efficient human-robot collaboration with robust plan recognition and trajectory prediction. *IEEE Robotics and Automation Letters*, 5 (2):2602–2609, 2020.

[2] J. Li, H. Ma, and M. Tomizuka. Conditional generative neural system for probabilistic trajectory prediction. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6150–6156. IEEE, 2019.

[3] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 365–371. IEEE, 2011.

[4] A. Abuduweili and C. Liu. Robust nonlinear adaptation algorithms for multitask prediction networks. *International Journal of Adaptive Control and Signal Processing*, 35(3):314–341, 2021.

[5] L. Wang, Y. Hu, L. Sun, W. Zhan, M. Tomizuka, and C. Liu. Transferable and adaptable driving behavior prediction. *arXiv preprint arXiv:2202.05140*, 2022.

[6] A. Abuduweili, S. Li, and C. Liu. Adaptable human intention and trajectory prediction for human-robot collaboration. *arXiv preprint arXiv:1909.05089*, 2019.

[7] Y. Cheng, W. Zhao, C. Liu, and M. Tomizuka. Human motion prediction using semi-adaptable neural networks. In *2019 American Control Conference (ACC)*, pages 4884–4890, 2019. doi: 10.23919/ACC.2019.8814980.

[8] A. Bajcsy, A. Siththaranjan, C. J. Tomlin, and A. D. Dragan. Analyzing human models that adapt online. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2754–2760. IEEE, 2021.

[9] W. Si, T. Wei, and C. Liu. Agen: Adaptable generative prediction networks for autonomous driving. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 281–286. IEEE, 2019.

[10] P. Kothari, D. Li, Y. Liu, and A. Alahi. Motion style transfer: Modular low-rank adaptation for deep motion forecasting. In *Conference on Robot Learning*, pages 774–784. PMLR, 2023.

[11] A. Blum. On-line algorithms in machine learning. *Online algorithms: the state of the art*, pages 306–325, 2005.

[12] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. D. Stefano. Real-time self-adaptive deep stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 195–204, 2019.

[13] A. Abuduweili and C. Liu. Robust online model adaptation by extended kalman filter with exponential moving average and dynamic multi-epoch strategy. In *Learning for Dynamics and Control*, pages 65–74. PMLR, 2020.

[14] S. Bhasin, R. Kamalapurkar, H. T. Dinh, and W. E. Dixon. Robust identification-based state derivative estimation for nonlinear systems. *IEEE Transactions on Automatic Control*, 58(1): 187–192, 2012.

[15] C. Paleologu, J. Benesty, and S. Ciochina. A robust variable forgetting factor recursive least-squares algorithm for system identification. *IEEE Signal Processing Letters*, 15:597–600, 2008.

[16] H. C. Barron. Neural inhibition for continual learning and memory. *Current opinion in neurobiology*, 67:85–94, 2021.

[17] J. Butepage, M. J. Black, D. Kragic, and H. Kjellstrom. Deep representation learning for human motion prediction and classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6158–6166, 2017.

[18] S. Lefèvre, D. Vasquez, and C. Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1–14, 2014.

[19] A. S. Weigend. *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018.

[20] H. H. Sohrab. *Basic real analysis*, volume 231. Springer, 2003.

[21] R. C. Dorf. *Modern control system*. 2008.

[22] T. Roughgarden. *Beyond the worst-case analysis of algorithms*. Cambridge University Press, 2021.

[23] N. E. Du Toit and J. W. Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2011.

[24] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1-3):159–178, 1992.

[25] D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431, 1979.

[26] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[27] A. Rudenko, T. P. Kucner, C. S. Swaminathan, R. T. Chadalavada, K. O. Arras, and A. J. Lilienthal. Thör: Human-robot navigation data collection and accurate motion trajectories dataset. *IEEE Robotics and Automation Letters*, 5(2):676–682, 2020.

[28] R. Liu and C. Liu. Human motion prediction using adaptable recurrent neural networks and inverse kinematics. *IEEE Control Systems Letters*, 5(5):1651–1656, 2020.

[29] J. Colyar and J. Halkias. Us highway 101 dataset. *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.

[30] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.