PRE: AN IN-CONTEXT RESIDUAL ESTIMATION PARADIGM FOR PLUG-AND-PLAY TIME-SERIES FORECAST CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series foundation models (TSFMs) are gaining popularity for their strong zero-shot forecasting capabilities and ease of deployment. However, recurring task-specific errors and distribution shifts create a need for adaptation during deployment, as frequent fine-tuning can be costly and requires careful hyperparameter selection and domain expertise. To address this challenge, we propose *Pretrained Residual Estimator* (PRE), which introduces a plug-and-play in-context learning (ICL) paradigm for post-hoc forecast correction. By conditioning on labeled historical examples, PRE infers residual corrections without task-specific online parameter updates. To support large-scale ICL pretraining, we synthesize diverse residuals and propose a two-stage pretraining strategy, first training PRE on synthetic residuals and then refining it on real residuals. PRE is independent of the base forecaster's internal architecture and supports variable context lengths and forecast horizons. Extensive evaluations demonstrate that PRE consistently improves the forecasting performance of unseen TSFMs on unseen tasks during its training and substantially outperforms supervised residual adapters, highlighting the potential of ICL as a generalizable paradigm for post-hoc forecast correction.

1 INTRODUCTION

Time-series foundation models (TSFMs) have shifted time-series forecasting from fitting task-specific models toward large-scale pretraining (Woo et al., 2024). By training on diverse time-series corpora, TSFMs can achieve strong zero-shot performance on unseen forecasting tasks (Das et al., 2024; Ansari et al., 2025). Their ability to transfer across tasks makes them attractive as out-of-the-box forecasters, particularly when task-specific data or model-tuning expertise are limited (Ansari et al., 2024; Khwaja et al., 2026).

Despite this progress, strong zero-shot performance does not ensure that a TSFM fully captures the characteristics of every downstream task. During online deployment, mismatches between a pretrained TSFM and task-specific dynamics can lead to recurring errors across successive forecasts (Dai et al., 2026). Task-specific fine-tuning offers a potential remedy (Qiao et al., 2025; Zhao et al., 2025), but it requires careful hyperparameter selection and model-tuning expertise (Tomar et al., 2025), and frequently updating large TSFMs online is computationally demanding (Lee et al., 2025). Distribution shifts common in real-world time series can further complicate adaptation, as changes in the underlying dynamics may introduce new error patterns (Liu et al., 2022; Kim et al., 2025). These challenges motivate efficient methods for improving forecasts while preserving the convenience of out-of-the-box deployment.

To address this need, recent online adaptation methods improve frozen TSFMs through lightweight calibration, ensemble forecasting, or residual correction (Kim et al., 2025; Lee et al., 2025; Dai et al., 2026). However, these approaches still rely on task-specific supervised updates of auxiliary models during deployment. The need for frequent and stable updating favors compact adaptation modules, often linear, which constrain their capacity to capture complex nonlinear relationships (Huang et al., 2026). In contrast, in-context learning (ICL) remains largely unexplored for forecast adaptation, despite its ability to construct task-specific regressors from labeled context examples without task-specific parameter updates (Akyürek et al., 2023; Hollmann et al., 2025). Historical residuals are

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

particularly suitable for such contextual supervision, as they are informative for a forecaster's recent error patterns and have long been used to inform subsequent estimation and correction (Firmino et al., 2015; Hyndman & Athanasopoulos, 2021). This raises a natural question: **Can historical forecasting experience serve as in-context supervision for generalizable forecast correction?**

We answer this question with a *Pretrained Residual Estimator* (PRE), a plug-and-play model that casts post-hoc forecast correction as an in-context residual regression problem. Through pretraining across diverse forecasting tasks, PRE acquires transferable residual estimation capabilities. During deployment, PRE adapts its correction through contextual inference over available forecasting experience, and estimates a residual distribution to refine the base forecast.

Our contributions are summarized as follows:

- We introduce a plug-and-play ICL paradigm for forecast correction. By constructing labeled examples from historical forecasting information, PRE adapts residual predictions through contextual inference without task-specific online training.
- To alleviate the data scarcity in large-scale pretraining, we develop a residual synthesis scheme that generates diverse and interpretable error patterns, and propose a two-stage pretraining strategy that combines synthetic and real residuals.
- Extensive evaluations demonstrate that PRE substantially improves the performance of base forecasters, generalizes consistently across unseen forecasting tasks and forecasters, and clearly outperforms existing supervised adaptation methods.

2 PRELIMINARIES

2.1 FOUNDATION MODELS FOR TIME-SERIES FORECASTING

Unlike conventional forecasters fitted individually to each dataset, TSFMs are pretrained on large, diverse time-series corpora. A single TSFM can perform zero-shot forecasting on unseen tasks and, in some settings, outperform models trained specifically on the target datasets (Woo et al., 2024).

Paradigm of TSFMs. For clarity, we first introduce the forecasting task in the univariate setting. Let $x_s \in \mathbb{R}$ denote the observation at time step s. At the current time step t, a forecaster $f(\cdot)$ receives a context $\boldsymbol{x}_t$ of length L and produces an H-step forecast $\hat{\boldsymbol{y}}_t^{\mathrm{B}}$:

$$\hat{\boldsymbol{y}}_t^{\mathrm{B}} = f(\boldsymbol{x}_t), \quad \boldsymbol{x}_t = [x_{t-L+1}, \ldots, x_t]^\top \in \mathbb{R}^L, \quad \boldsymbol{y}_t = [x_{t+1}, \ldots, x_{t+H}]^\top \in \mathbb{R}^H. \tag{1}$$

Here, $\hat{\boldsymbol{y}}_t^{\mathrm{B}} = [\hat{x}_{t+1|t}^{\mathrm{B}}, \ldots, \hat{x}_{t+H|t}^{\mathrm{B}}]^\top \in \mathbb{R}^H$ is the forecast for the target $\boldsymbol{y}_t$. $x_s \in \mathbb{R}$ is the observation at time step s, and $\hat{x}_{s|t}^{\mathrm{B}}$ is the forecast of x_s based on the information available up to time step t. For multivariate forecasting, $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ can be represented as matrices with time and variable dimensions.

A TSFM $f_\theta(\cdot)$ learns this forecasting mapping through pretraining on a large-scale dataset $\mathcal{D}$ containing time series from diverse sources:

$$\theta^\star \in \arg\min_\theta \mathbb{E}_{(\boldsymbol{x}_t, \boldsymbol{y}_t)\sim\mathcal{D}} \left[\ell \left(f_\theta(\boldsymbol{x}_t), \boldsymbol{y}_t \right) \right], \tag{2}$$

where $\theta^\star$ denotes the learned model parameters and ℓ is the model-specific training loss. After pretraining, $f_{\theta^\star}(\cdot)$ can be directly applied to unseen forecasting tasks with its parameters frozen. Many TSFMs provide predictive quantiles for point forecasting, uncertainty quantification, and risk analysis (Ansari et al., 2025; Liu et al., 2025):

$$\hat{\boldsymbol{y}}_t^{\mathrm{B},(\alpha)} = f_{\theta^\star}^{(\alpha)}(\boldsymbol{x}_t) \in \mathbb{R}^H, \quad \alpha \in (0,1), \tag{3}$$

where $f_{\theta^\star}^{(\alpha)}(\cdot)$ denotes the model's output at level α.

Existing TSFMs and Limitations. Recent works have explored diverse architectures and pretraining strategies to improve generalization across forecasting tasks. TimesFM adopts a decoder-only architecture (Das et al., 2024), Chronos formulates forecasting through tokenized probabilistic

prediction (Ansari et al., 2024; 2025), and Moirai employs a masked encoder-based architecture with unified training across heterogeneous time series (Woo et al., 2024). More recent TSFMs further scale model capacity and broaden the range of supported forecasting settings (Liu et al., 2025; Auer et al., 2025; Khwaja et al., 2026).

To improve TSFM performance on specific downstream tasks, recent studies have investigated specialized fine-tuning strategies (Qiao et al., 2025; Zhao et al., 2025). However, fine-tuning is not an out-of-the-box solution, as it requires task-specific optimization and hyperparameter selection (Tomar et al., 2025). Full-parameter fine-tuning can also incur substantial computational costs (Li & Zhu, 2026) and risk overfitting when task-specific data are limited (Park et al., 2025). Moreover, real-world time series often exhibit distribution shifts as their underlying dynamics evolve (Liu et al., 2022). Offline fine-tuning falls short of adapting timely to distribution shifts, while repeatedly updating large TSFMs online can be computationally demanding (Lee et al., 2025).

2.2 Online Adaptation for TSFM Forecasting

Formulation of Online Adaptation. Online adaptation aims to improve forecasting performance by accumulating available information as the data stream unfolds (Lee et al., 2025; Kim et al., 2025). Generally, these adaptation methods can be formulated as:

$$\hat{\boldsymbol{y}}_t = g_{\phi_t}\left(\hat{\boldsymbol{y}}_t^{\mathrm{B}}, \boldsymbol{c}_t\right), \tag{4}$$

where $g_{\phi_t}(\cdot)$ is an adaptation module whose parameters ϕ_t are typically updated during deployment, and $\boldsymbol{c}_t$ is additional causally available information, such as the input context, summaries of recent observations, or historical error statistics (Hoi et al., 2021).

Existing Residual Correction Methods. Several recent methods specifically estimate residual corrections for frozen forecasters. δ-Adapter (Liang et al., 2026) uses a small MLP with bounded outputs to learn conservative corrections to base forecasts. COSA (Im & Kwon, 2026) combines base forecasts with statistics of recently observed targets to produce linear residual corrections. ORCA (Dai et al., 2026) estimates residuals from the input context and base forecast using a linear adapter, and uses historical errors to compute the weights for combining base and corrected forecasts. Despite their different implementations, these adapters are fundamentally supervised learning models. Their need for frequent and stable parameter updates necessitates the use of compact model architectures, which limits their expressive capacity (Raghu et al., 2017).

An alternative approach to correcting forecasts using residuals is to explicitly include historical residuals as model inputs, as they directly characterize how a forecaster has erred under recent data distributions (Huang et al., 2026). Exploiting such error information has a long history in state estimation (Kalman, 1960) and has also proved effective in task-specific forecasting (Zhou et al., 2025; Huang et al., 2026). However, some studies on residual correction have explored this approach but reported no additional gains (Dai et al., 2026). One possible explanation is that their simple architectures limit the ability to capture relationships between forecast errors and the scenarios in which they arise.

3 Methodology

3.1 In-Context Residual Estimation with Forecast Memory

ICL enables pretrained models to construct task-specific regressors directly from input–output examples, without requiring task-specific parameter updates (Akyürek et al., 2023; Hollmann et al., 2025). This paradigm provides a natural mechanism for exploiting historical forecasting experience: since previously observed residuals contain informative error patterns (Zhang, 2003), they can serve as supervision within the context. This observation motivates us to formulate residual estimation as an ICL problem, allowing the pretrained model to adapt its residual predictions to the evolving data distribution and forecasting behavior, as illustrated in Figure 1.

To formalize the in-context regression, we first introduce two concepts. For a forecast origin s, we define the *forecast scenario* $(\boldsymbol{x}_s, \hat{\boldsymbol{y}}_s^{\mathrm{B}})$, which characterizes both the data condition and the base forecaster's behavior under the given input. Once $\boldsymbol{y}_s$ becomes fully observed, its residual trajectory

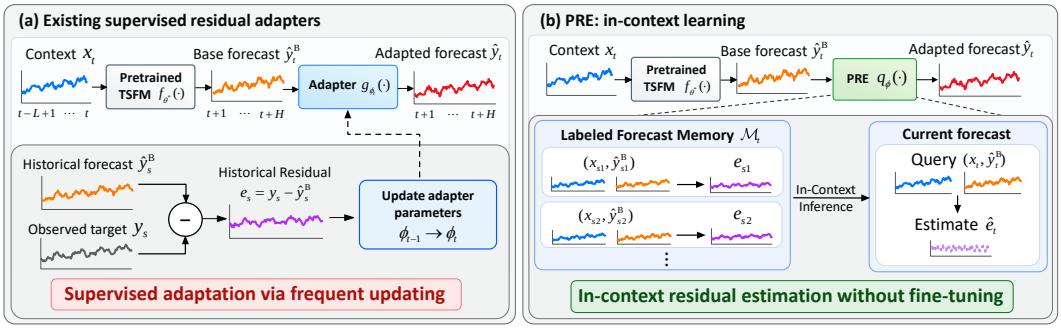

Figure 1: Schematic of supervised residual adapters and PRE. **(a)** Existing adapters learn task-specific corrections through repeated supervised parameter updates during deployment. **(b)** PRE treats historical scenario–residual pairs as labeled examples and the current forecast scenario as the query, performing in-context residual estimation without deployment-time parameter updates.

can be computed as $\boldsymbol{e}_s = \boldsymbol{y}_s - \hat{\boldsymbol{y}}_s^{\mathrm{B}} = [e_{s+1|s}, \ldots, e_{s+H|s}]^\top \in \mathbb{R}^H$. To keep the in-context examples compact and reduce redundancy from overlapping forecast targets, we define the *forecast memory* for the current time step t as the K most recent fully observed examples with non-overlapping forecast targets:

$$\mathcal{M}_t = \left\{ \left(\boldsymbol{x}_{s_k}, \hat{\boldsymbol{y}}_{s_k}^{\mathrm{B}}, \boldsymbol{e}_{s_k} \right) \right\}_{k=1}^{K}, \qquad s_k = t - kH, \quad k = 1, \ldots, K. \tag{5}$$

Each element of $\mathcal{M}_t$ pairs a historical forecast scenario with its observed residual, providing a labeled in-context example of how the base forecaster behaves under a particular data condition.

Residual estimation is formulated as a probabilistic regression problem, with the forecast memory $\mathcal{M}_t$ providing in-context examples that pair historical forecast scenarios with their residuals. The current residual $\boldsymbol{e}_t$ is modeled as a conditional distribution $p\left(\boldsymbol{e}_t \mid \boldsymbol{x}_t, \hat{\boldsymbol{y}}_t^{\mathrm{B}}, \mathcal{M}_t\right)$. Since the true conditional distribution is unknown, we propose a *Pretrained Residual Estimator* (PRE), parameterized by ϕ, to approximate it. PRE performs residual estimation in an ICL manner: the historical scenario–residual pairs in $\mathcal{M}_t$ serve as labeled examples, and the current forecast scenario $\left(\boldsymbol{x}_t, \hat{\boldsymbol{y}}_t^{\mathrm{B}}\right)$ serves as the query. Through pretraining across diverse forecasting tasks, PRE learns to infer the residual distribution of the query from relevant historical examples:

$$q_\phi\left(\boldsymbol{e}_t \mid \boldsymbol{x}_t, \hat{\boldsymbol{y}}_t^{\mathrm{B}}, \mathcal{M}_t\right) \approx p\left(\boldsymbol{e}_t \mid \boldsymbol{x}_t, \hat{\boldsymbol{y}}_t^{\mathrm{B}}, \mathcal{M}_t\right). \tag{6}$$

A point estimate $\hat{\boldsymbol{e}}_t$ can then be obtained as, e.g., the median of the estimated residual distribution and used to refine the base forecast:

$$\hat{\boldsymbol{y}}_t = \hat{\boldsymbol{y}}_t^{\mathrm{B}} + \hat{\boldsymbol{e}}_t. \tag{7}$$

3.2 Architecture of PRE

PRE implements in-context residual estimation through labeled historical examples and a current query. Specifically, in each memory element $\left(\boldsymbol{x}_{s_k}, \hat{\boldsymbol{y}}_{s_k}^{\mathrm{B}}, \boldsymbol{e}_{s_k}\right)$, the forecast scenario $\left(\boldsymbol{x}_{s_k}, \hat{\boldsymbol{y}}_{s_k}^{\mathrm{B}}\right)$ serves as the input of an in-context example, and the residual $\boldsymbol{e}_{s_k}$ serves as its label. The current forecast scenario $\left(\boldsymbol{x}_t, \hat{\boldsymbol{y}}_t^{\mathrm{B}}\right)$ forms an unlabeled query. PRE encodes these labeled examples and the query into a shared latent space and performs context-conditioned inference of the residual distribution $q_\phi(\cdot)$ in equation 6.

Latent Encoding. Given the current query $\left(\boldsymbol{x}_t, \hat{\boldsymbol{y}}_t^{\mathrm{B}}\right)$ and labeled examples in $\mathcal{M}_t$, we first use patch embedding (Nie et al., 2023) to convert the contexts, base forecasts, and historical residual labels into patch tokens. Transformer encoders (Vaswani et al., 2017) then model dependencies across patches within each sequence. Denoting each encoding branch by $\mathcal{E}_a = \mathrm{Enc}_a \circ \mathrm{PatchEmb}_a$ for $a \in \{x, y, e\}$, we obtain

$$\boldsymbol{X}_s = \mathcal{E}_x(\boldsymbol{x}_s), \qquad \boldsymbol{Y}_s = \mathcal{E}_y(\hat{\boldsymbol{y}}_s^{\mathrm{B}}), \qquad \boldsymbol{E}_s = \mathcal{E}_e(\boldsymbol{e}_s). \tag{8}$$

Here, $\boldsymbol{X}_s \in \mathbb{R}^{N_x \times d}$ represents the context, and $\boldsymbol{Y}_s, \boldsymbol{E}_s \in \mathbb{R}^{N_y \times d}$ represent the base forecast and residual label, respectively. N_x and N_y are the numbers of context and forecast patches, and d

is the latent dimension. The context and forecast representations are computed for both historical examples and the current query, whereas residual representations are available only for the labeled examples in $\mathcal{M}_t$. The same patching configuration is used for the base forecast and residual to preserve their temporal alignment.

To characterize how the base forecaster maps the observed context to the future, we use cross-attention, with $\boldsymbol{Y}_s$ as queries and $\boldsymbol{X}_s$ as keys and values. Each forecast patch aggregates relevant context information, yielding the scenario representation

$$\boldsymbol{S}_s = \mathrm{CrossAttn}\left(Q = \boldsymbol{Y}_s, K = \boldsymbol{X}_s, V = \boldsymbol{X}_s\right) \in \mathbb{R}^{N_y \times d}. \tag{9}$$

The same embedding, encoding, and cross-attention modules are shared across all historical examples and the current query, placing their scenario representations in a common latent space for subsequent in-context matching.

In-Context Residual Interaction. The residual label of a historical example is informative only in relation to the forecast scenario under which it was observed. PRE therefore performs in-context inference by matching the current query against historical labeled examples and aggregating their residual information according to scenario relevance.

Specifically, for the p-th forecast patch, we use the current scenario representation $\boldsymbol{S}_{t,p}$ as the query, the historical scenario representations $\{\boldsymbol{S}_{s_k,p}\}_{k=1}^{K}$ as keys, and their corresponding residual-label representations $\{\boldsymbol{E}_{s_k,p}\}_{k=1}^{K}$ as values:

$$\boldsymbol{R}_{t,p} = \mathrm{CrossAttn}\left(Q = \boldsymbol{S}_{t,p}, K = \{\boldsymbol{S}_{s_k,p}\}_{k=1}^{K}, V = \{\boldsymbol{E}_{s_k,p}\}_{k=1}^{K}\right) \in \mathbb{R}^{d}. \tag{10}$$

This operation constitutes the core in-context interaction in PRE: residual labels from historical examples are retrieved and aggregated according to their relevance to the current forecast query. A recency bias is further incorporated to account for the temporal distance between each historical example and the current query. Stacking the aggregated evidence vectors yields $\boldsymbol{R}_t \in \mathbb{R}^{N_y \times d}$. We then fuse $\boldsymbol{R}_t$ with the current scenario representation $\boldsymbol{S}_t$ through element-wise addition and apply a Transformer encoder to model dependencies across forecast patches:

$$\boldsymbol{Z}_t = \mathrm{Enc}_r\left(\boldsymbol{S}_t + \boldsymbol{R}_t\right) \in \mathbb{R}^{N_y \times d}. \tag{11}$$

The resulting $\boldsymbol{Z}_t$ therefore combines the current query with residual information inferred from the labeled in-context examples.

Probabilistic Residual Estimation. We estimate the current residuals by modeling a conditional marginal distribution for each forecast step, a common practice in TSFMs (Ansari et al., 2025; Liu et al., 2025), thereby avoiding explicit parameterization of the full H-step joint distribution. Using a bin-based parameterization (Müller et al., 2022), a probabilistic output head maps $\boldsymbol{Z}_t$ to $q_{\phi,h}\big(\cdot \mid \boldsymbol{x}_t, \hat{\boldsymbol{y}}_t^{\mathrm{B}}, \mathcal{M}_t\big)$ for $h = 1, \ldots, H$. Each marginal incorporates information inferred from the in-context examples through $\boldsymbol{Z}_t$. Details of the probabilistic output design are provided in Appendix C.1. We take the median of each estimated residual distribution as the point correction:

$$\hat{e}_{t+h|t} = \mathrm{Median}\big[q_{\phi,h}\big(\cdot \mid \boldsymbol{x}_t, \hat{\boldsymbol{y}}_t^{\mathrm{B}}, \mathcal{M}_t\big)\big], \quad h = 1, \ldots, H. \tag{12}$$

The resulting estimates form $\hat{\boldsymbol{e}}_t$ and are used to refine the base forecast.

3.3 Training Method

Residual Synthesis and Two-Stage Training. To acquire transferable in-context residual inference capability, PRE requires a large collection of data during pretraining. Relying solely on real residuals from base forecasters limits the diversity of error patterns exposed to PRE. We therefore introduce an efficient residual synthesis scheme to enrich pretraining data with diverse interpretable error patterns. Specifically, we add a synthetic perturbation $\boldsymbol{\delta}_t$ to a real residual $\boldsymbol{e}_t = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t^{\mathrm{B}}$, obtaining the synthetic residual $\tilde{\boldsymbol{e}}_t$ and its corresponding augmented forecast $\tilde{\boldsymbol{y}}_t^{\mathrm{B}}$:

$$\tilde{\boldsymbol{e}}_t = \boldsymbol{e}_t + \boldsymbol{\delta}_t, \qquad \tilde{\boldsymbol{y}}_t^{\mathrm{B}} = \boldsymbol{y}_t - \tilde{\boldsymbol{e}}_t = \hat{\boldsymbol{y}}_t^{\mathrm{B}} - \boldsymbol{\delta}_t. \tag{13}$$

We construct $\boldsymbol{\delta}_t$ as a convex combination of multiple residual components:

$$\boldsymbol{\delta}_t = \sum_{j=1}^{J} w_j \boldsymbol{\delta}_t^{(j)}, \qquad w_j \geq 0, \qquad \sum_{j=1}^{J} w_j = 1, \tag{14}$$

where $\boldsymbol{\delta}_t^{(j)}$ is the j-th sampled residual component and w_j is its mixture weight. We design eight types of interpretable components $\boldsymbol{\delta}_t^{(j)}$, including persistent bias, bias that gradually increases along the forecast horizon, missed local dynamics caused by overly smooth forecasts, and importantly, white noise without predictability. For each channel, we randomly sample a set of residual components to maintain consistent error characteristics within that channel while introducing rich error patterns across channels and datasets. Details for each residual component are provided in Appendix C.2.

To effectively exploit both synthetic and real residuals, we design a two-stage training strategy. In the first stage, PRE is trained on augmented forecasts and their corresponding synthetic residuals to learn broad, interpretable error patterns. In the second stage, the synthetic perturbations are removed. PRE is further trained on the unmodified forecasts and their realized residuals from the same forecasters in the first stage, to capture more subtle and complex error structures.

Training Objective. Our training objective combines a negative log-likelihood (NLL) term commonly used in probabilistic forecasting with an improvement regularizer:

$$\mathcal{L} = \mathcal{L}_{\mathrm{NLL}} + \lambda \mathcal{L}_{\mathrm{imp}}, \tag{15}$$

where $\lambda > 0$ controls the weight of the regularizer. The NLL term trains PRE to estimate the conditional residual distributions:

$$\mathcal{L}_{\mathrm{NLL}} = -\frac{1}{H} \sum_{h=1}^{H} \log q_{\phi,h}(e_{t+h|t} | \boldsymbol{x}_t, \hat{\boldsymbol{y}}_t^{\mathrm{B}}, \mathcal{M}_t). \tag{16}$$

The improvement regularizer $\mathcal{L}_{\mathrm{imp}}$ reflects the actual benefit of residual correction by comparing the refined and base forecasts:

$$\mathcal{L}_{\mathrm{imp}} = \mathrm{Softplus}\left(\frac{\mathcal{L}^{\mathrm{R}} - \mathcal{L}^{\mathrm{B}}}{\tau}\right), \quad \mathcal{L}^{\mathrm{B}} = \frac{1}{H}\sum_{h=1}^{H} \left|e_{t+h|t}\right|, \quad \mathcal{L}^{\mathrm{R}} = \frac{1}{H}\sum_{h=1}^{H} \left|e_{t+h|t} - \hat{e}_{t+h|t}\right|. \tag{17}$$

Here, $\mathcal{L}^{\mathrm{B}}$ and $\mathcal{L}^{\mathrm{R}}$ are the mean absolute errors of the base and refined forecasts, respectively, and $\tau > 0$ is a temperature parameter. $\mathcal{L}_{\mathrm{imp}}$ increases with $\mathcal{L}^{\mathrm{R}} - \mathcal{L}^{\mathrm{B}}$, thereby explicitly forcing the estimator to generate residual corrections with actual forecasting improvements. The regularizer hyperparameters are provided in Appendix B.3.

4 Experiments

4.1 General Setup

Benchmark and Data Splits. We evaluate PRE in both in-domain and zero-shot settings on TIME, a benchmark tailored for TSFM evaluation (Qiao et al., 2026). TIME covers 50 datasets and 98 forecasting tasks across eight domains, and provides predefined test and non-test segments for each dataset. Among the 50 datasets, we retain 39 datasets with sufficient sequence length for our experiments and split them within each domain into two groups. For datasets in the first group, the non-test segments are used for pretraining and validation, and the test segments are used for in-domain evaluation. Datasets in the second group are excluded from pretraining. Their non-test segments are used only to construct forecast memory during inference, and their test segments are used for zero-shot evaluation, thereby assessing PRE's cross-task generalization capability. Details of TIME and our selection criteria are provided in Appendix A.2.

Training and Evaluation Protocol. Forecast memories for PRE training are generated exclusively from three lightweight forecasters: AutoARIMA (Hyndman & Khandakar, 2008), AutoTheta (Fiorucci et al., 2016), and DLinear (Zeng et al., 2023). Following the two-stage strategy in Section 3.3, the first stage augments the residuals produced by these three forecasters with synthetic error patterns, and the second stage uses their unmodified residuals. PRE's hyperparameters and optimization settings are provided in Appendix B.3. During evaluation, we refine the forecasts of 10 TSFMs that are never used during PRE training, thereby assessing PRE's cross-model generalization capability. Details of the evaluated TSFMs are provided in Appendix A.1. For both evaluation settings defined in Section 4.1, we report the mean absolute scaled error (MASE) and its relative reduction over the corresponding base TSFMs.

4.2 Overall Correction Performance

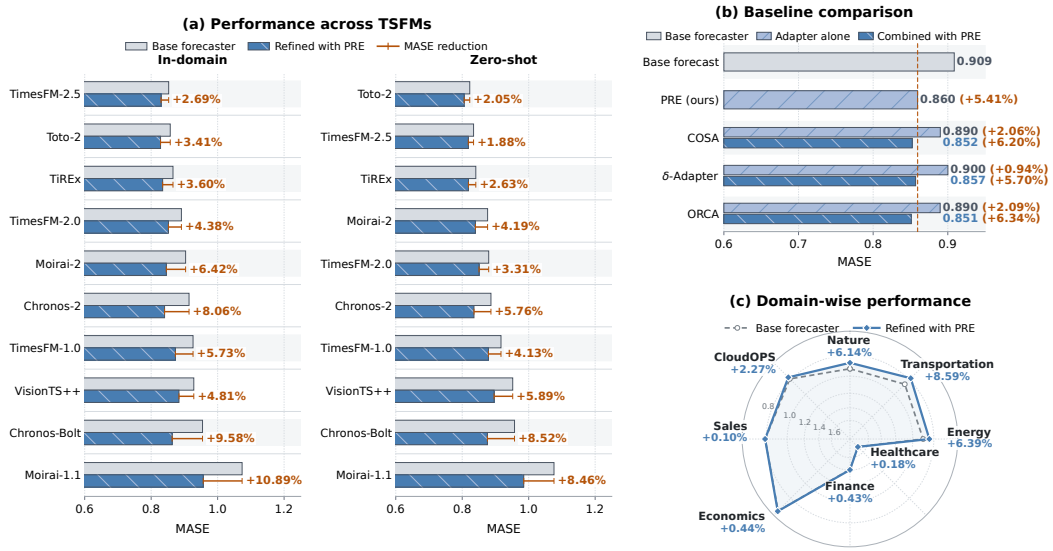

Figure 2: Forecast correction performance of PRE across different TSFMs and domains on the TIME benchmark. Percentages denote relative MASE reductions over base TSFM forecasts. **(a)** Performance before and after PRE correction for 10 TSFMs in the in-domain and zero-shot settings. Models are ordered from top to bottom by their base-forecast MASE. **(b)** Comparison with baselines, with MASE averaged across all evaluated datasets in TIME and the 10 TSFMs. PRE's plug-and-play design allows it to be readily combined with existing baselines to achieve better correction performance. **(c)** Domain-wise comparison of base and PRE-refined forecasts. The radar chart uses a reversed radial scale, with lower MASE farther from the center.

Figure 2(a) displays forecasting performance before and after applying PRE. PRE improves the performance of all 10 unseen TSFMs in both evaluation settings. For the strongest base forecasters, PRE reduces MASE by 2.69% for TimesFM-2.5 in-domain and by 2.05% for Toto-2 under zero-shot evaluation. Larger gains are observed for relatively weaker forecasters: Moirai-1.1 achieves MASE reductions of 10.89% in-domain and 8.46% zero-shot, and Chronos-Bolt improves by 9.58% and 8.52%, respectively. These consistent improvements demonstrate that PRE can transfer its correction capability to both unseen forecasters and unseen tasks without task-specific supervised fine-tuning. Detailed results are provided in Appendix D.1. Overall, forecasters with higher base MASE tend to benefit more from PRE. One possible explanation is that weaker forecasters leave more pronounced and predictable error patterns in their residuals. For stronger forecasters, the remaining error patterns may be subtler and thus harder to disentangle reliably. Appendix D.6 presents visual examples of PRE's forecast correction, including cases where base forecasts are overly smooth, exhibit temporal lag, or are already accurate.

4.3 Comparison with Supervised Residual Adapters

We compare PRE with three supervised residual adapters, ORCA (Dai et al., 2026), δ-Adapter (Liang et al., 2026), and COSA (Im & Kwon, 2026), which can be applied on top of different base forecasters. As discussed in Section 2.2, these adapters rely on task-specific supervised updating during deployment. As shown in Figure 2(b), PRE apparently outperforms these standalone adapters, achieving a relative MASE reduction of 5.41%. To maintain efficient updating, the supervised adapters rely on lightweight correction modules at the expense of model capacity. Although such modules can efficiently capture clear trends and periodic patterns in conventional long-term forecasting benchmarks (Zhou et al., 2021; Zeng et al., 2023), their limited capacity makes it harder to capture complex nonlinear and time-varying dynamics (Liu et al., 2023), which are prevalent in TIME (Qiao et al., 2026).

Besides, PRE's plug-and-play design allows it to be readily combined with these baselines. To investigate their complementarity, we treat each TSFM together with a baseline adapter as a composite forecaster and apply PRE to refine its outputs. All three combinations outperform both PRE and the corresponding baseline alone. These results suggest that PRE's in-context correction paradigm is complementary to existing supervised online adaptation methods. Detailed results for each baseline are provided in Appendix D.2.

4.4 Domain-wise Performance

Figure 2(c) reports PRE's domain-wise performance, with results averaged over all datasets within each domain. PRE achieves consistent improvements across all eight domains, with notable gains in Nature, Energy, Transportation, and CloudOPS. More modest gains are observed in Finance, Economics, Healthcare, and Sales. One possible reason is that forecasting errors in these domains are driven by unrecorded external factors. For example, Healthcare and Sales tasks are substantially subject to external factors such as public policies, human behavior, and market interventions (Kraemer et al., 2020; Stock & Watson, 2002), whereas PRE does not explicitly incorporate such exogenous information. Extending forecast memory to incorporate cross-channel dependencies and exogenous covariates may therefore further improve performance in such domains. Complete domain-wise results are provided in Appendix D.3.

4.5 Ablation Studies

Contribution of Residual Labels. To assess the contribution of residual labels, we compare the default PRE with two ablated variants. In the first, we set residuals in $\mathcal{M}_t$ to zero for the default PRE during inference. In the second, we retrain PRE with the same architecture but set residuals to zero during both training and inference. As shown in Table 1, removing residuals only during inference reduces the gains from $+5.99\%$ to $+0.64\%$ in-domain and from $+4.71\%$ to $+0.79\%$ zero-shot. This substantial drop indicates that the correction capability learned by PRE relies on historical residual labels. When residuals are removed during both training and inference, PRE achieves an in-domain gain of $+2.93\%$, but the gain falls to $+1.56\%$ in the zero-shot setting. These results demonstrate that historical residual labels contribute substantially to forecasting improvements and are particularly important for PRE's generalization performance. Detailed results are provided in Appendix D.4.1.

Table 1: Contribution of historical residual inputs. We report average MASE and its relative reductions (Gain) over 10 TSFMs. The best results are shown in **bold**.

Residual Usage		In-domain		Zero-shot	
Training	**Inference**	**MASE**	**Gain**	**MASE**	**Gain**
✓	✓	**0.860**	**+5.99%**	**0.859**	**+4.71%**
✓	✗	0.909	+0.64%	0.894	+0.79%
✗	✗	0.888	+2.93%	0.887	+1.56%

Contribution of Residual Synthesis. To evaluate the contribution of residual synthesis and the training strategy, we investigate four settings: training on synthetic residuals only, real residuals only, a one-stage mixture of both, and our default two-stage strategy. Table 2 reports average performance over all 10 TSFMs (All-10) and the stronger and weaker halves (Top-5 and Bottom-5). Training on synthetic residuals alone yields gains of 4.95% in-domain and 3.59% zero-shot. The four strategies perform similarly on Bottom-5, suggesting that synthetic residuals characterize generalizable error patterns. One-stage mixing improves All-10 results over either source alone, indicating their complementarity. The two-stage strategy further improves All-10 and Top-5 performance, with a more pronounced advantage on Top-5. Detailed results are provided in Appendix D.4.2.

4.6 Forecast Memory Size and Deployment Cost

We train PRE with forecast memory sizes $K \in \{1, 4, 8, 16, 32, 64\}$ and measure the additional inference cost using a batch size of 1 on an AMD EPYC 9334 CPU and an NVIDIA RTX 5880 Ada Generation GPU. Figure 3(a) shows that increasing K generally improves forecasting accuracy at

Table 2: Performance of PRE under different training strategies. Comparison of PRE training strategies. "All-10" covers all 10 evaluated TSFMs; "Top-5" and "Bottom-5" denote the stronger and weaker halves, respectively. "Mixed (One-stage)" trains the estimator on both synthetic and real residuals at once. "Mixed (Two-stage)" trains first on synthetic and then on real residuals. Best results are shown in **bold**.Best results are shown in **bold**.

Setting	Training Strategy	All-10	Top-5	Bottom-5
In-domain	Mixed (Two-stage)	**0.860 (+5.99%)**	**0.838 (+4.10%)**	**0.883 (+7.84%)**
	Mixed (One-stage)	0.863 (+5.61%)	0.845 (+3.32%)	**0.883 (+7.84%)**
	Synthetic Only	0.869 (+4.95%)	0.849 (+2.85%)	0.890 (+7.02%)
	Real Only	0.866 (+5.34%)	0.845 (+3.28%)	0.887 (+7.36%)
Zero-shot	Mixed (Two-stage)	**0.859 (+4.71%)**	**0.826 (+2.82%)**	0.893 (+6.57%)
	Mixed (One-stage)	0.864 (+4.18%)	0.836 (+1.67%)	**0.892 (+6.60%)**
	Synthetic Only	0.869 (+3.59%)	0.840 (+1.28%)	0.900 (+5.85%)
	Real Only	0.866 (+3.92%)	0.835 (+1.79%)	0.898 (+6.00%)

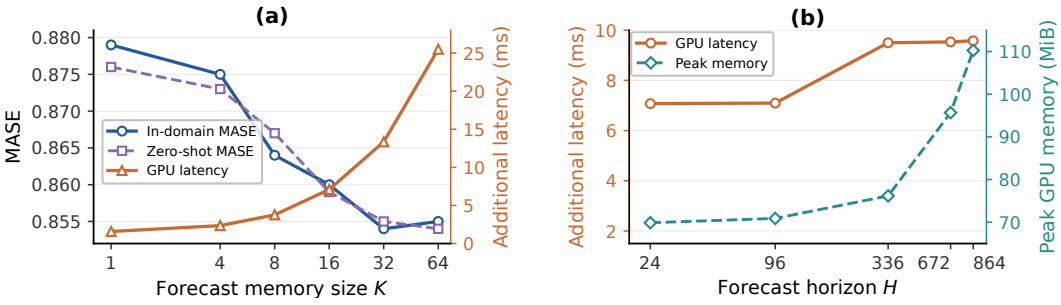

Figure 3: Forecast memory size and GPU deployment cost of PRE. **(a)** MASE and additional GPU inference latency across different K with $H = 96$. **(b)** Additional GPU inference time and peak additional GPU memory usage across different H with $K = 16$.

the cost of higher inference latency. This suggests that a longer memory provides additional useful information for residual correction, but this benefit does not grow indefinitely. In particular, doubling K from 32 to 64 brings little improvement since some early trajectories may be less informative for characterizing current error evolution, while the inference latency continues to increase. We therefore adopt $K = 16$ as a practical trade-off between forecasting performance and computational cost. With $K = 16$, Figure 3(b) shows that PRE incurs modest additional inference overhead on the evaluated GPU, even for long forecast horizons. Detailed performance of each K and additional inference cost are provided in Appendices D.4.3 and D.5, respectively.

5 Conclusion

In this work, we present PRE, a plug-and-play model that formulates post-hoc forecast correction as an in-context residual regression. Using labeled historical scenario–residual examples, PRE adapts its corrections through in-context inference without deployment-time parameter updates. To support large-scale ICL pretraining with limited real residual data, we design interpretable residual priors to generate diverse synthetic residuals and adopt a two-stage training strategy that first trains PRE on synthetic residuals and then refines it on real residuals. Experiments across 10 TSFMs unseen during PRE training demonstrate consistent forecasting improvements and generalization to unseen tasks, and substantially outperforms supervised residual adapters. Although our evaluation focus on TSFMs, PRE does not depend on the base forecaster's internal architecture and can be applied to any other forecasters. PRE currently estimates residuals independently for each channel and does not explicitly capture cross-channel dependencies or incorporate exogenous covariates. Incorporating such information and developing richer synthetic residual priors are promising directions for future work.

AI USE STATEMENT

The core ideas of this work were proposed by the authors, with AI tools used to assist in identifying and reviewing related work. The model architecture was designed by the authors, and the code implementation was developed with AI assistance and reviewed by the authors. AI tools also assisted in organizing raw experimental results into figures and tables for the manuscript, with the data verified by the authors. During manuscript preparation, AI tools were used to polish the language of the authors' initial draft. The authors reviewed and verified the AI-assisted content and take full responsibility for the final manuscript, code, and reported results.

ETHICS STATEMENT

This study uses publicly available time-series datasets from the TIME benchmark (Qiao et al., 2026). Although PRE improves forecasting accuracy on average in our experiments, it does not guarantee improvements for every prediction. Practical applications should therefore include appropriate validation and human oversight. Pretraining PRE requires substantial computational resources, which may make reproducing the training process difficult for researchers with limited resources. To improve accessibility, we provide the pretrained model and accompanying code, enabling other researchers to apply PRE without repeating the pretraining process.

REPRODUCIBILITY STATEMENT

We provide both the checkpoint of PRE and the code in the supplementary materials. To help readers get started, we include a `demo.ipynb` notebook that guides them through environment setup and synthetic data generation. The notebook also demonstrates PRE's forecast correction on example tasks.

REFERENCES

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=0g0X4H8yN4I.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=gerNCVqqtR.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025. URL https://arxiv.org/abs/2510.15821.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. TiRex: Zero-shot forecasting across long and short horizons with enhanced in-context learning. In *Advances in Neural Information Processing Systems*, 2025.

Xilin Dai, Yiding Liu, Hongjie Xia, Yifan Hu, Zewei Dong, Jiang-Ming Yang, and Qiang Xu. Learning the context of errors: Black-box online adaptation of time series foundation models. *arXiv preprint arXiv:2606.14222*, 2026.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine*

Learning, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167. PMLR, 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Jose A Fiorucci, Tiago R Pellegrini, Francisco Louzada, Fotios Petropoulos, and Anne B Koehler. Models for optimising the theta method and their relationship to state space models. *International journal of forecasting*, 32(4):1151–1161, 2016.

Paulo Renato A. Firmino, Paulo Salgado Gomes de Mattos Neto, and Tiago A. E. Ferreira. Error modeling approach to improve time series forecasters. *Neurocomputing*, 153:242–254, 2015.

Steven C. H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025. doi: 10.1038/s41586-024-08328-6.

Xiannan Huang, Shen Fang, Shuhan Qiu, Chengcheng Yu, Jiayuan Du, and Chao Yang. TEFL: Prediction-residual-guided rolling forecasting for multi-horizon time series, 2026. URL https://arxiv.org/abs/2602.22520.

Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 3 edition, 2021.

Rob J Hyndman and Yeasmin Khandakar. Automatic time series forecasting: the forecast package for r. *Journal of statistical software*, 27(3):1–22, 2008.

Jeonghwan Im and Hyuk-Yoon Kwon. COSA: Context-aware output-space adapter for test-time adaptation in time series forecasting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=L7Z5wBMPrW.

Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era, 2026. URL https://arxiv.org/abs/2605.20119.

HyunGi Kim, Siwon Kim, Jisoo Mok, and Sungroh Yoon. Battling the non-stationarity in time series forecasting via test-time adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17868–17876, 2025. doi: 10.1609/aaai.v39i17.33965. URL https://ojs.aaai.org/index.php/AAAI/article/view/33965.

Moritz U. G. Kraemer, Chia-Hung Yang, Bernardo Gutierrez, Chieh-Hsi Wu, Brennan Klein, David M. Pigott, et al. The effect of human mobility and control measures on the COVID-19 epidemic in china. *Science*, 368(6490):493–497, 2020. doi: 10.1126/science.abb4218.

Thomas L. Lee, William Toner, Rajkarn Singh, Artjom Joosen, and Martin Asenov. Lightweight online adaption for time series foundation model forecasts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 33736–33764. PMLR, 2025. URL https://proceedings.mlr.press/v267/lee25ag.html.

Yuze Li and Wei Zhu. TRACE: Time series parameter efficient fine-tuning. *Neurocomputing*, 664:132098, 2026. doi: 10.1016/j.neucom.2025.132098. URL https://www.sciencedirect.com/science/article/pii/S0925231225027705.

Daojun Liang, Qi Li, Yinglong Wang, Jing Chen, Hu Zhang, Xiaoxiao Cui, Qizheng Wang, and Shuo Li. The forecast after the forecast: A post-processing shift in time series. In *The Fourteenth International Conference on Learning Representations*, 2026.

Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Silvio Savarese, Doyen Sahoo, Caiming Xiong, and Junnan Li. Moirai 2.0: When less is more for time series forecasting. *arXiv preprint arXiv:2511.11698*, 2025.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/4054556fcaa934b0bf76da52cf4f92cb-Abstract-Conference.html.

Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with koopman predictors. In *Advances in Neural Information Processing Systems*, volume 36, pp. 12271–12290, 2023.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

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*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Young-Jin Park, François G. Germain, Jing Liu, Ye Wang, Toshiaki Koike-Akino, Gordon Wichern, Navid Azizan, Christopher R. Laughman, and Ankush Chakrabarty. Probabilistic forecasting for building energy systems using time-series foundation models. *Energy and Buildings*, 348:116446, 2025. doi: 10.1016/j.enbuild.2025.116446. URL https://www.merl.com/publications/TR2026-030.

Zhongzheng Qiao, Chenghao Liu, Yiming Zhang, Ming Jin, Quang Pham, Qingsong Wen, Ponnuthurai Suganthan, Xudong Jiang, and Savitha Ramasamy. Multi-scale finetuning for encoder-based time series foundation models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.neurips.cc/paper_files/paper/2025/hash/20563b8508ba42e1b688d922e926ee26-Abstract-Conference.html.

Zhongzheng Qiao, Sheng Pan, Anni Wang, Viktoriya Zhukova, Yong Liu, Xudong Jiang, Qingsong Wen, Mingsheng Long, Ming Jin, and Chenghao Liu. It's TIME: Towards the next generation of time series forecasting benchmarks. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=79TgfXHbsK.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2847–2854, 2017.

Lefei Shen, Mouxiang Chen, Xu Liu, Han Fu, Xiaoxue Ren, Jianling Sun, Zhuo Li, and Chenghao Liu. VisionTS++: Cross-modal time series foundation model with continual pre-trained vision backbones, 2025. URL https://arxiv.org/abs/2508.04379.

James H. Stock and Mark W. Watson. Macroeconomic forecasting using diffusion indexes. *Journal of Business & Economic Statistics*, 20(2):147–162, 2002. doi: 10.1198/073500102317351921.

Shivani Tomar, Seshu Tirupathi, Radu Marinescu, Elizabeth M. Daly, and Ivana Dusparic. AT4TS: Autotune for time series foundation models. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=U54YyLn8MX.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper/7181-attention-is-all-you-need.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning*

Research, pp. 53140–53164. PMLR, 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

G. Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003. doi: 10.1016/S0925-2312(01)00702-0.

Lifan Zhao, Yanyan Shen, Zhaoyang Liu, Xue Wang, and Jiaji Deng. Less is more: Unlocking specialization of time series foundation models via structured pruning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.nips.cc/paper_files/paper/2025/hash/facaf651f417d3e4afaf23c579e701e6-Abstract-Conference.html.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Yifan Zhou, Yibo Wang, and Chao Shang. IRNN: Innovation-driven recurrent neural network for time-series data modeling and prediction, 2025. URL https://arxiv.org/abs/2505.05916.

A Datasets and Baselines

A.1 TSFM Baselines

We evaluate 10 TSFMs from the TimesFM (Das et al., 2024), Chronos (Ansari et al., 2024; 2025), Moirai (Woo et al., 2024; Liu et al., 2025), Toto (Khwaja et al., 2026), TiRex (Auer et al., 2025), and VisionTS (Shen et al., 2025) families. Table 3 summarizes their characteristics.

Table 3: Overview of the TSFMs used in our experiments. Model names are linked to their checkpoints. In the "Input" column, "Uni." indicates models that support only univariate forecasting, whereas "Multi." indicates additional support for multivariate forecasting. "Output" refers to the model's output type. "Parameter scales" lists all released model sizes, with the size used in our experiments shown in **bold**.

TSFM	Architecture	Input	Output	Parameter scales
Chronos-2	Encoder-only	Multi.	Quantile	**120M**
Chronos-Bolt	Encoder–decoder	Uni.	Quantile	9M, 21M, 48M, **205M**
TimesFM-2.5	Decoder-only	Uni.	Quantile	**230M**
TimesFM-2.0	Decoder-only	Uni.	Quantile	**500M**
TimesFM-1.0	Decoder-only	Uni.	Quantile	**200M**
Moirai-2	Decoder-only	Uni.	Quantile	**11M**
Moirai-1.1	Masked encoder	Multi.	Distribution	14M, **91M**, 311M
Toto-2	Decoder-only	Multi.	Quantile	4M, **22M**, 313M, 1B, 2.5B
TiRex	xLSTM	Uni.	Quantile	**35M**
VisionTS++	Visual MAE	Multi.	Quantile	**115M**, 333M

To accommodate differences in the maximum context length supported by each model, we use a context length of 512 for all evaluations. All TSFMs generate forecasts in the univariate setting. For models with quantile outputs, we use the median quantile as the point forecast; for models that generate distributions, we sample 100 forecast trajectories and independently take their median at each horizon step to obtain the point forecast. The TSFM parameters remain frozen throughout the entire experimental process.

A.2 TIME BENCHMARK

TIME is a task-centric benchmark tailored for TSFM evaluation, designed to reduce benchmark contamination and to align forecasting configurations with real-world applications (Qiao et al., 2026). It curates fresh time-series datasets from government portals, industrial and academic sources, open-access repositories, and forecasting competitions. Its construction pipeline combines automatic quality screening with human decision-making. Forecast horizons are selected based on each dataset's sampling frequency and application context. Across the benchmark, forecast horizons range from 3 to 864 time steps.

TIME comprises 50 datasets and 98 forecasting tasks across eight application domains. Each forecasting task is defined by a dataset and a forecast horizon. Following TIME's convention, we treat the same underlying data sampled at different frequencies as separate datasets. Each dataset may contain multiple input groups, each identified by the `input_name` field in the benchmark manifest. This field follows the format `TIME:{dataset}/{frequency}/{series}`. For example, `TIME:CPHL/15T/NRSROT` identifies the NRSROT station series in the CPHL dataset at a 15-minute sampling interval.

A.3 DATASET SPLITS

As described in Section 4.1, we divide TIME datasets into in-domain and zero-shot sets. For a domain containing n datasets, we randomly assign $\lfloor n/2 \rfloor$ of them to the zero-shot set, reserved for zero-shot evaluation. The remaining datasets form the in-domain set for PRE training and evaluation. Our main experiments use a context length of 512 and require forecast memory for adaptation. We therefore exclude 11 of TIME's 50 datasets whose available context lengths are shorter than 512 observations. As a result, our evaluation covers 39 datasets and 85 forecasting tasks: 21 datasets and 46 tasks for in-domain evaluation, and 18 datasets and 39 tasks for zero-shot evaluation. The dataset assignments are listed in Table 4.

Table 4: Assignments of datasets from TIME in our experiments. The sampling frequencies of each dataset are shown in parentheses. "#" denotes the total number of datasets in each domain.

Domain	#	In-domain datasets	Zero-shot datasets
Nature	16	CPHL(15T), CPHL(30T), CPHL(H), Coastal_T_S(5T), Coastal_T_S(15T), Coastal_T_S(20T), Coastal_T_S(H), NE_China_Wind(H), SG_PM25(H)	SG_Weather(D), Water_Quality_Darwin(15T), current_velocity(5T), current_velocity(10T), current_velocity(15T), current_velocity(20T), current_velocity(H)
Energy	4	Australia_Solar(H), EWELD_Load(15T)	OpenElectricity_NEM(5T), epf_electricity_price(H)
Transportation	3	Finland_Traffic(15T), Port_Activity(D)	SG_Carpark(15T)
Healthcare	1	–	ECDC_COVID(D)
Finance	3	Crypto(D), Oil_Price(B)	US_Term_Structure(B)
Economics	3	Uncertainty_1M(M)	Commodity_Import(M), Commodity_Production(M)
Sales	4	Supply_Chain_Location(D), Vehicle_Sales(M)	Online_Retail_2_UCI(D), Supply_Chain_Customer(D)
CloudOPS	5	MetroPT-3(5T), azure2019_D(5T), azure2019_U(5T)	Smart_Manufacturing(H), azure2019_I(5T)
Total	**39**	**21 datasets**	**18 datasets**

B Training and Evaluation Details

B.1 Training and Evaluation Forecasters

We construct PRE's training memories using three base forecasters: AutoTheta (Fiorucci et al., 2016), DLinear (Zeng et al., 2023), and AutoARIMA (Hyndman & Khandakar, 2008). We adopt the two-stage training strategy described in Section 3.3. In the first stage, we train PRE on augmented forecasts and their synthetic residuals. In the second stage, we continue training on unmodified forecasts and their real residuals. We evaluate the same PRE across the 10 TSFMs listed in Table 7. None of these TSFMs is exposed to PRE during training. All experiments in our work were conducted using random seed 2026.

B.2 Model Configuration

Table 5 summarizes PRE's architecture. Patch embeddings use sequence values, adjacent differences, and validity masks as inputs. Current and historical scenarios share the same context and forecast encoding modules. Unavailable residuals are zero-filled and masked; when no valid historical residuals are available, PRE returns a zero correction.

Table 5: Default model configuration of PRE.

Hyperparameter	**Value**
Context length	512
Historical memory size K	16
Context patch length / stride	16 / 8
Forecast and residual patch length / stride	4 / 4
Hidden dimension d	288
Attention heads	8
Feed-forward dimension	1152
Encoder depths: context / forecast / residual	3 / 1 / 2
Fusion encoder depth	3
Dropout	0.1
Activation	GELU
Distribution bins	64
Number of parameters	10,679,584

B.3 Training Hyperparameters and Optimization

Table 6 reports the optimization settings for the two training stages. Synthetic training memories are generated offline before Stage 1 and remain fixed throughout that stage. Stage 2 starts from the best Stage-1 checkpoint, with the optimizer, learning-rate scheduler, and early-stopping state reset. PRE parameters are optimized only during offline pretraining. During evaluation, the same pretrained PRE checkpoint is kept frozen across all tasks and base forecasters. In each epoch, we sample up to 4,096 training examples for each combination of base forecaster and forecasting task. Each pretraining example consists of a forecast memory, a current forecast scenario, and the current residual target. Historical scenario–residual pairs form the labeled context. The current scenario is supplied as an unlabeled query, and its residual is used only to compute the training objective. Batch sizes are adjusted dynamically based on the forecast horizon, with a maximum of 256 examples per GPU. Both stages use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We apply linear warm-up over the first 5% of optimizer updates, followed by cosine learning-rate decay to zero. The regularization weight λ and temperature τ in Eq. equation 15 are both fixed at 0.1 throughout training. We evaluate validation performance after every epoch using the geometric mean of normalized validation MASE of the three base forecasters. Early stopping uses a minimum of eight epochs and a patience of ten epochs.

B.4 Evaluation Metrics and Aggregation

We use MASE as our primary evaluation metric. MASE is widely adopted in TSFM studies, as its scale independence makes it suitable for cross-task aggregation. Following TIME (Qiao et al.,

Table 6: Hyperparameters for PRE's two-stage training. Stage 1 uses synthetic residuals, and Stage 2 uses real residuals.

Hyperparameter	**Stage 1**	**Stage 2**
Optimizer	AdamW	AdamW
Initial learning rate	2×10^{-4}	1×10^{-4}
Weight decay	0.01	0.01
Training epochs	20	20
Maximum batch size per GPU	256	256
Number of GPUs	4	4
Warm-up ratio	5%	5%
Learning-rate decay	Cosine	Cosine
Gradient clipping norm	1.0	1.0
Regularization weight λ	0.1	0.1
Temperature τ	0.1	0.1

2026), we aggregate task-level MASE scores using the geometric mean, which treats multiplicative improvements and degradations symmetrically and reduces the influence of extreme scores compared with arithmetic averaging.

MASE Aggregation. Let $q_{m,k}$ denote the MASE of TSFM m on task k, and let $\mathcal{T}_S$ denote the task set for in-domain ($S = \text{ID}$) or zero-shot ($S = \text{ZS}$) evaluation. We first aggregate across tasks for each TSFM, then across the $M = 10$ TSFMs:

$$Q_{m,S} = \left(\prod_{k \in \mathcal{T}_S} q_{m,k} \right)^{1/|\mathcal{T}_S|}, \qquad Q_S = \left(\prod_{m=1}^{M} Q_{m,S} \right)^{1/M}. \tag{18}$$

Here, $Q_{m,S}$ is the aggregated MASE of TSFM m across tasks in evaluation setting S, and Q_S is the aggregated MASE across all $M = 10$ TSFMs in the same setting. The in-domain and zero-shot settings include 46 and 39 tasks, respectively. Their overall score is

$$Q_{\text{all}} = Q_{\text{ID}}^{46/85} Q_{\text{ZS}}^{39/85}, \tag{19}$$

which gives equal weight to all 85 tasks.

Relative MASE Reduction. We aggregate the MASE scores of base and PRE-refined forecasts separately using the same procedure. At each reporting level, let Q^{Base} and Q^{PRE} denote the corresponding aggregated scores, respectively. The relative MASE reduction is then computed as

$$\text{Gain} = \left(1 - \frac{Q^{\text{PRE}}}{Q^{\text{Base}}} \right) \times 100\%. \tag{20}$$

B.5 Hardware and Software Environment

All experiments were conducted on a server equipped with two AMD EPYC 9334 CPUs (64 physical cores and 128 hardware threads in total), 256 GB of system memory, and four NVIDIA RTX 5880 Ada Generation GPUs with 48 GB of memory per GPU. The software environment uses Ubuntu 22.04.4 LTS, Python 3.10.20, and PyTorch 2.3.1 with CUDA 12.1 and cuDNN 8.9.2.

C Method Details

C.1 Probabilistic Output

Following Müller et al. (2022), we use the bin-based density parameterization to model the residual distribution. Specifically, we divide the range of possible residual values into B intervals, called bins, and predict the probability that the residual falls into each bin. Given the fused representation $\boldsymbol{Z}_t$ discussed in Section 3.2, the output head applies RMS normalization and a shared linear projection, mapping each forecast patch of length $P = 4$ to $P \times B$ logits, where $B = 64$ is the number

of distribution bins. These logits are then reshaped and passed through a softmax to obtain the bin probabilities $\boldsymbol{\pi}_{t,h} \in \mathbb{R}^B$ for each forecast step h.

In practice, we apply binning in a transformed residual space. Let z denote a normalized residual. We model its transformed value $u = \operatorname{asinh}(z)$, which compresses large residual magnitudes and preserves their signs. We define equally spaced reference boundaries $a_0, \ldots, a_B$ over $[-6, 6]$. Within each bin, the density is uniform:

$$q^{(u)}_{\phi,h}(u) = \frac{\pi_{t,h,b}}{a_b - a_{b-1}}, \quad u \in (a_{b-1}, a_b], \quad b = 2, \ldots, B-1. \tag{21}$$

To accommodate residuals beyond the reference range, the first and last bins use one-sided exponential densities on $(-\infty, a_1]$ and (a_{B-1}, ∞), carrying probability masses $\pi_{t,h,1}$ and $\pi_{t,h,B}$, respectively. Both tails use a rate of $\log 2/\Delta$, where $\Delta = 12/B$ is the reference bin width. The resulting distribution covers the entire real line without clipping extreme residuals.

For each forecast step, we compute the median in the transformed space and apply the inverse transformation $\sinh$ to obtain the normalized residual estimate. We then restore the residual estimate to the original scale and add it to the base forecast.

C.2 Residual Synthesis

We construct synthetic residuals by adding structured perturbations to real residuals, as described in Section 3.3. For each channel, let $f_{t,h} = \hat{y}^{\mathrm{B}}_{t+h|t}$ denote the base forecast at step h. We denote the last observation, mean, standard deviation, and context volatility by ℓ_t, μ_t, σ_t, and v_t, respectively. Here, v_t is the mean absolute first-order difference of the context window. The designs of the eight residual perturbation components are as follows.

Persistent bias. This component introduces a constant shift across the forecast horizon:

$$\delta^{(1)}_{t,h} = b, \quad h = 1, \ldots, H,$$

where $b \in \{-1, +1\}$ determines the perturbation direction. The same definition of b is used throughout the following components.

Horizon-dependent drift. This component introduces a power-law drift along the forecast horizon:

$$\delta^{(2)}_{t,h} = b \left(\frac{h}{H} \right)^{\gamma},$$

where γ controls the curvature of the drift.

Amplitude distortion. This component modifies the forecast amplitude relative to a context-dependent reference level. We define the reference level as

$$a_t = \eta \ell_t + (1 - \eta)\mu_t,$$

and construct the residual perturbation as

$$\delta^{(3)}_{t,h} = b(a_t - f_{t,h}),$$

where η controls the contribution of the last observation and context mean.

Phase shift. This component introduces temporal phase shifts, causing the forecast trajectory to appear advanced or delayed relative to the underlying dynamics. Let $f^{\mathrm{lag}}_{t,h}$ denote the shifted forecast trajectory. The perturbation is

$$\delta^{(4)}_{t,h} = b(f^{\mathrm{lag}}_{t,h} - f_{t,h}).$$

Oversmoothing. This component simulates overly smooth forecasts that miss local dynamics. We recursively generate a smoothed forecast trajectory:

$$s_{t,0} = \ell_t, \quad s_{t,h} = \alpha f_{t,h} + (1 - \alpha) s_{t,h-1},$$

where α controls the degree of smoothing. The perturbation is defined as

$$\delta^{(5)}_{t,h} = s_{t,h} - f_{t,h}.$$

High-frequency omission. This component introduces discrepancies associated with high-frequency variations:

$$\delta_{t,h}^{(6)} = f_{t,h} - \mathrm{MA}_w(\boldsymbol{f}_t)_h,$$

where MA_w denotes a moving average with window size w.

State-dependent bias. This component models errors that depend on the current forecasting state. It considers both the deviation from historical statistics and the local changing rate of the forecast trajectory. Let

$$z_{t,h} = \frac{f_{t,h} - \mu_t}{\sigma_t}, \quad d_{t,h} = \frac{f_{t,h} - f_{t,h-1}}{v_t}, \quad f_{t,0} = \ell_t,$$

where $z_{t,h}$ measures the state deviation, and $d_{t,h}$ measures the local variation. The perturbation is

$$\delta_{t,h}^{(7)} = b\sigma_t \left[(1-\eta)\,\mathrm{sign}(z_{t,h})(|z_{t,h}| - \theta)_+ + \eta \tanh(d_{t,h})\right],$$

where θ controls the level threshold and η balances the contributions of state deviation and local variation.

White noise. We additionally include an unpredictable noise component:

$$\delta_{t,h}^{(8)} = \epsilon_{t+h}, \quad \epsilon_\tau \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1).$$

For each channel, a subset of perturbation components is randomly selected and combined:

$$\boldsymbol{\delta}_t = \sum_{j=1}^{J} w_j \boldsymbol{\delta}_t^{(j)}, \quad w_j \geq 0, \quad \sum_{j=1}^{J} w_j = 1.$$

The perturbations are normalized and rescaled according to the local context scale to control their magnitudes. The resulting synthetic residual is then obtained as

$$\tilde{\boldsymbol{e}}_t = \boldsymbol{e}_t + \boldsymbol{\delta}_t.$$

D Experimental Details

D.1 Detailed Correction Results

Table 7 reports the complete per-forecaster results in Section 4.2, including the MASE before and after applying PRE and the corresponding relative improvement under both in-domain and zero-shot evaluation. PRE improves all 10 unseen TSFMs in both settings, with average MASE reductions of 5.99% in-domain and 4.71% zero-shot.

D.2 Detailed Baseline Comparison

Tables 8 and 9 provide the per-forecaster results for the baseline comparison in Section 4.3. We compare PRE with three representative supervised residual adaptation methods, including ORCA (Dai et al., 2026), δ-Adapter (Liang et al., 2026), and COSA (Im & Kwon, 2026). PRE consistently outperforms these standalone baselines across all 10 TSFMs, achieving an average MASE reduction of 5.41%. Benefiting from its plug-and-play design, PRE can also be readily combined with these existing adaptation methods to further refine their forecasts. All three combinations improve upon both PRE and these baselines, with PRE + ORCA achieving the largest average gain of 6.34%. These results further support the complementarity between PRE's ICL paradigm and existing supervised adaptation methods.

D.3 Detailed Domain-Wise Performance

Tables 10 and 11 provide the dataset-wise results corresponding to the domain-level analysis in Section 4.4. Consistent with the domain-level observations, PRE achieves positive average gains across all eight domains. Among these domains, Transportation achieves the largest average improvement

Table 7: Forecasting performance of 10 unseen TSFMs before and after applying PRE. MASE is reported, with lower values indicating better performance. Gain denotes the relative MASE reduction achieved by PRE over the corresponding base forecast, computed as $(\text{Base} - \text{PRE})/\text{Base} \times 100\%$. Average results are computed using the geometric mean across TSFMs. PRE consistently improves the forecasting performance of all 10 TSFMs across both in-domain and zero-shot settings.

Forecaster	In-domain			Zero-shot		
	Base↓	PRE↓	Gain↑	Base↓	PRE↓	Gain↑
Toto-2	0.857	0.828	+3.41%	0.823	0.806	+2.05%
TimesFM-2.5	0.853	0.830	+2.69%	0.834	0.818	+1.88%
TiRex	0.866	0.835	+3.60%	0.841	0.819	+2.63%
TimesFM-2.0	0.891	0.852	+4.38%	0.880	0.851	+3.31%
Moirai-2	0.904	0.846	+6.42%	0.876	0.840	+4.19%
Chronos-2	0.914	0.840	+8.06%	0.886	0.835	+5.76%
TimesFM-1.0	0.926	0.873	+5.73%	0.917	0.879	+4.13%
VisionTS++	0.928	0.884	+4.81%	0.952	0.896	+5.89%
Chronos-Bolt	0.955	0.863	+9.58%	0.957	0.876	+8.52%
Moirai-1.1	1.074	0.957	+10.89%	1.076	0.985	+8.46%
Geo. Mean	0.915	0.860	+5.99%	0.901	0.859	+4.71%

Table 8: Performance of PRE and supervised residual adapters across 10 TSFMs. MASE and the relative MASE reduction over base forecasts are reported.

TSFM	Base	PRE	COSA	δ-Adapter	ORCA
Toto-2	0.841	0.818 (+2.79%)	0.836 (+0.61%)	0.837 (+0.56%)	0.829 (+1.43%)
TimesFM-2.5	0.844	0.825 (+2.32%)	0.836 (+0.98%)	0.839 (+0.65%)	0.833 (+1.27%)
TiREx	0.854	0.827 (+3.16%)	0.843 (+1.28%)	0.846 (+1.00%)	0.834 (+2.36%)
TimesFM-2.0	0.886	0.851 (+3.89%)	0.871 (+1.69%)	0.882 (+0.40%)	0.878 (+0.91%)
Moirai-2	0.891	0.843 (+5.41%)	0.872 (+2.19%)	0.877 (+1.55%)	0.858 (+3.65%)
Chronos-2	0.901	0.838 (+7.01%)	0.880 (+2.33%)	0.894 (+0.82%)	0.885 (+1.82%)
TimesFM-1.0	0.922	0.876 (+5.00%)	0.907 (+1.59%)	0.915 (+0.69%)	0.909 (+1.34%)
VisionTS++	0.939	0.889 (+5.30%)	0.916 (+2.48%)	0.928 (+1.16%)	0.929 (+1.09%)
Chronos-Bolt	0.956	0.869 (+9.09%)	0.925 (+3.18%)	0.943 (+1.35%)	0.920 (+3.73%)
Moirai-1.1	1.075	0.970 (+9.78%)	1.029 (+4.23%)	1.062 (+1.18%)	1.040 (+3.22%)
Geo. Mean	0.909	0.860 (+5.41%)	0.890 (+2.06%)	0.900 (+0.94%)	0.890 (+2.09%)

of $+8.59\%$, followed by Energy ($+6.39\%$) and Nature ($+6.14\%$). Within these domains, datasets such as `Finland_Traffic`, `OpenElectricity_NEM`, and `Coastal_T_S` achieve substantial gains of $+19.28\%$, $+11.47\%$, and $+13.58\%$, respectively. These datasets exhibit strong recurring temporal structures, and their underlying temporal dynamics can be largely inferred from historical

Table 9: Performance of PRE combined with supervised residual adapters across 10 TSFMs. MASE and the relative MASE reduction over base forecasts are reported.

TSFM	Base	PRE	PRE + COSA	PRE + δ-Adapter	PRE + ORCA
Toto-2	0.841	0.818 (+2.79%)	0.815 (+3.10%)	0.816 (+2.96%)	0.813 (+3.33%)
TimesFM-2.5	0.844	0.825 (+2.32%)	0.819 (+2.92%)	0.822 (+2.66%)	0.817 (+3.17%)
TiREx	0.854	0.827 (+3.16%)	0.822 (+3.73%)	0.823 (+3.64%)	0.815 (+4.56%)
TimesFM-2.0	0.886	0.851 (+3.89%)	0.845 (+4.63%)	0.850 (+4.01%)	0.848 (+4.25%)
Moirai-2	0.891	0.843 (+5.41%)	0.836 (+6.16%)	0.838 (+5.90%)	0.830 (+6.84%)
Chronos-2	0.901	0.838 (+7.01%)	0.832 (+7.71%)	0.837 (+7.07%)	0.832 (+7.70%)
TimesFM-1.0	0.922	0.876 (+5.00%)	0.868 (+5.80%)	0.874 (+5.18%)	0.871 (+5.54%)
VisionTS++	0.939	0.889 (+5.30%)	0.879 (+6.38%)	0.886 (+5.68%)	0.886 (+5.69%)
Chronos-Bolt	0.956	0.869 (+9.09%)	0.863 (+9.76%)	0.865 (+9.49%)	0.852 (+10.81%)
Moirai-1.1	1.075	0.970 (+9.78%)	0.952 (+11.45%)	0.967 (+10.06%)	0.955 (+11.13%)
Geo. Mean	0.909	0.860 (+5.41%)	0.852 (+6.20%)	0.857 (+5.70%)	0.851 (+6.34%)

observations. As a result, the residual patterns revealed by $\mathcal{M}_t$ are more stable and informative, enabling PRE to provide more effective corrections.

In contrast, the improvements are relatively smaller in domains such as Healthcare, Sales, Finance, and Economics, where temporal evolution can be strongly influenced by unobserved external factors, including policy changes, human behavior, and market conditions (Kraemer et al., 2020; Stock & Watson, 2002). Nevertheless, PRE maintains robust performance even in these challenging scenarios: only four datasets show negative gains in the in-domain setting, and zero-shot evaluation produces almost no degradation. Besides, the negative effects remain marginal (e.g., `Port_Activity`: -0.01%, `Vehicle_Sales`: -0.11%). These results demonstrate that PRE provides effective and reliable residual correction across diverse domains.

Table 10: Dataset-wise in-domain performance on TIME. "Avg. Gain" reports the average relative MASE reduction across ten base forecasters on each dataset. "Improved TSFMs" reports the number of TSFMs with positive gain on this dataset.

Domain	Dataset	Avg. Gain	Improved TSFMs
Nature	CPHL (15T)	+5.42%	10/10
Nature	CPHL (30T)	+2.70%	10/10
Nature	CPHL (H)	+3.05%	9/10
Nature	Coastal_T_S (15T)	+8.69%	10/10
Nature	Coastal_T_S (20T)	+13.58%	10/10
Nature	Coastal_T_S (5T)	+9.27%	10/10
Nature	Coastal_T_S (H)	+8.45%	10/10
Nature	NE_China_Wind (H)	+0.15%	6/10
Nature	SG_PM25 (H)	+1.63%	10/10
Energy	Australia_Solar (H)	+5.17%	10/10
Energy	EWELD_Load (15T)	+8.59%	10/10
Transportation	Finland_Traffic (15T)	+19.28%	10/10
Transportation	Port_Activity (D)	-0.01%	2/10
Finance	Crypto (D)	-0.98%	2/10
Finance	Oil_Price (B)	+1.58%	6/10
Economics	Uncertainty_1M (M)	-0.46%	1/10
Sales	Supply_Chain_Location (D)	+0.27%	4/10
Sales	Vehicle_Sales (M)	-0.11%	0/10
CloudOPS	MetroPT-3 (5T)	+5.51%	10/10
CloudOPS	azure2019_D (5T)	+2.12%	9/10
CloudOPS	azure2019_U (5T)	+1.13%	4/10

D.4 Ablation Details

D.4.1 Contribution of In-Context Residual Labels

Tables 12 and 13 provide the per-forecaster results corresponding to the ablation study in Section 4.5. When historical residuals are available during training but removed at inference, the gains decrease to $+0.64\%$ in-domain and $+0.79\%$ zero-shot on average, with only limited improvements for most forecasters. When residuals are set to zero during both training and inference, the estimator retains average gains of $+2.93\%$ and $+1.56\%$, respectively, but remains substantially below the default PRE. These per-forecaster results further confirm that historical residual inputs make a substantial contribution to PRE's forecasting improvements, particularly in the zero-shot setting.

D.4.2 Contribution of Residual Synthesis

Tables 14, 15, and 16 provide the per-forecaster results corresponding to the residual-synthesis ablation study in Section 4.5. Training with real residuals only yields average gains of $+5.34\%$ in-domain and $+3.92\%$ zero-shot, while training with synthetic residuals alone achieves $+4.95\%$ and $+3.59\%$, respectively. The consistent improvements obtained with synthetic residuals across most forecasters further support the transferability of the error patterns generated by our residual synthesis scheme. Combining synthetic and real residuals leads to further improvements.

Table 11: Dataset-wise zero-shot performance on TIME. "Avg. Gain" reports the average relative MASE reduction across ten TSFMs on each dataset. "Improved TSFMs" reports the number of TSFMs with positive gain on each dataset.

Domain	Dataset	Avg. Gain	Improved TSFMs
Nature	SG_Weather (D)	+0.14%	4/10
Nature	Water_Quality_Darwin (15T)	+7.92%	10/10
Nature	current_velocity (10T)	+8.28%	10/10
Nature	current_velocity (15T)	+7.73%	10/10
Nature	current_velocity (20T)	+6.44%	10/10
Nature	current_velocity (5T)	+7.90%	10/10
Nature	current_velocity (H)	+5.81%	10/10
Energy	OpenElectricity_NEM (5T)	+11.47%	10/10
Energy	epf_electricity_price (H)	-0.04%	3/10
Transportation	SG_Carpark (15T)	+5.39%	9/10
Healthcare	ECDC_COVID (D)	+0.18%	3/10
Finance	US_Term_Structure (B)	+0.67%	4/10
Economics	Commodity_Import (M)	+0.21%	3/10
Economics	Commodity_Production (M)	+1.56%	6/10
Sales	Online_Retail_2_UCI (D)	0.00%	0/10
Sales	Supply_Chain_Customer (D)	+0.25%	6/10
CloudOPS	Smart_Manufacturing (H)	+0.04%	7/10
CloudOPS	azure2019_I (5T)	+2.44%	9/10

Table 12: Forecasting performance when historical context residual labels are replaced with zeros only at inference. MASE is reported. Average results are computed using the geometric mean across TSFMs.

TSFM	In-domain			Zero-shot		
	Base	Ablated PRE	Gain	Base	Ablated PRE	Gain
Toto-2	0.857	0.858	-0.10%	0.823	0.822	+0.07%
TimesFM-2.5	0.853	0.853	-0.06%	0.834	0.834	+0.07%
TiREx	0.866	0.866	-0.08%	0.841	0.840	+0.05%
TimesFM-2.0	0.891	0.889	+0.28%	0.880	0.880	-0.05%
Moirai-2	0.904	0.903	+0.09%	0.876	0.875	+0.15%
Chronos-2	0.914	0.894	+2.19%	0.886	0.862	+2.77%
TimesFM-1.0	0.926	0.923	+0.30%	0.917	0.911	+0.58%
VisionTS++	0.928	0.925	+0.39%	0.952	0.947	+0.53%
Chronos-Bolt	0.955	0.934	+2.18%	0.957	0.931	+2.80%
Moirai-1.1	1.074	1.061	+1.20%	1.076	1.067	+0.88%
Geo. Mean	0.915	0.909	+0.64%	0.901	0.894	+0.79%

D.4.3 Effect of Forecast Memory Size

Tables 17, 18, 19, 20, and 21 report the per-forecaster results for ablating the forecast memory size in Section 4.6. The average gains increase from $+3.89\%$ to $+6.63\%$ in-domain and from $+2.83\%$ to $+5.11\%$ zero-shot as K increases from 1 to 32. Doubling K from 32 to 64 yields diminishing marginal gains. These results demonstrate that increasing the forecast memory size can improve the correction performance within a certain range, whereas overly old memory provides limited value for the current forecast scenario.

D.5 Detailed Online Deployment Cost

We evaluate the online deployment cost of PRE on the server introduced in Appendix B. All experiments use a batch size of 1, a context length of 512, and the forecast memory $\mathcal{M}_t$ with $K = 16$. Timing starts after the base forecaster produces its forecast and ends when PRE outputs the residual estimate, thereby isolating the additional deployment overhead introduced by PRE. We also mea-

Table 13: Forecasting performance when historical context residual labels are replaced with zeros throughout pretraining and inference. MASE is reported. Average results are computed using the geometric mean across TSFMs.

TSFM	In-domain			Zero-shot		
	Base	Ablated PRE	Gain	Base	Ablated PRE	Gain
Toto-2	0.857	0.847	+1.21%	0.823	0.822	+0.04%
TimesFM-2.5	0.853	0.843	+1.11%	0.834	0.830	+0.45%
TiREx	0.866	0.855	+1.23%	0.841	0.838	+0.30%
TimesFM-2.0	0.891	0.875	+1.80%	0.880	0.873	+0.81%
Moirai-2	0.904	0.869	+3.78%	0.876	0.871	+0.65%
Chronos-2	0.914	0.863	+5.54%	0.886	0.855	+3.60%
TimesFM-1.0	0.926	0.907	+2.04%	0.917	0.905	+1.25%
VisionTS++	0.928	0.911	+1.85%	0.952	0.935	+1.80%
Chronos-Bolt	0.955	0.899	+5.84%	0.957	0.918	+4.16%
Moirai-1.1	1.074	1.023	+4.72%	1.076	1.049	+2.49%
Geo. Mean	0.915	0.888	+2.93%	0.901	0.887	+1.56%

Table 14: Forecasting performance when PRE is trained using only real residuals from the lightweight base forecasters. MASE is reported.

TSFM	In-domain			Zero-shot		
	Base	Ablated PRE	Gain	Base	Ablated PRE	Gain
Toto-2	0.857	0.838	+2.22%	0.823	0.816	+0.79%
TimesFM-2.5	0.853	0.836	+1.92%	0.834	0.823	+1.29%
TiREx	0.866	0.843	+2.62%	0.841	0.826	+1.80%
TimesFM-2.0	0.891	0.853	+4.22%	0.880	0.858	+2.44%
Moirai-2	0.904	0.855	+5.39%	0.876	0.853	+2.63%
Chronos-2	0.914	0.854	+6.61%	0.886	0.844	+4.77%
TimesFM-1.0	0.926	0.872	+5.87%	0.917	0.876	+4.46%
VisionTS++	0.928	0.883	+4.92%	0.952	0.898	+5.62%
Chronos-Bolt	0.955	0.880	+7.77%	0.957	0.887	+7.33%
Moirai-1.1	1.074	0.951	+11.47%	1.076	0.992	+7.78%
Geo. Mean	0.915	0.866	+5.34%	0.901	0.866	+3.92%

sure the additional memory usage incurred by PRE under both CPU and GPU execution. For each hardware setting and forecast horizon, we conduct 100 consecutive forecasts.

Table 15: Forecasting performance when PRE is trained using only synthetic residuals. MASE is reported.

TSFM	In-domain			Zero-shot		
	Base	Adapted	Gain	Base	Adapted	Gain
Toto-2	0.857	0.840	+1.99%	0.823	0.823	-0.09%
TimesFM-2.5	0.853	0.839	+1.60%	0.834	0.830	+0.53%
TiREx	0.866	0.848	+2.05%	0.841	0.831	+1.13%
TimesFM-2.0	0.891	0.860	+3.47%	0.880	0.861	+2.15%
Moirai-2	0.904	0.858	+5.08%	0.876	0.853	+2.66%
Chronos-2	0.914	0.853	+6.68%	0.886	0.846	+4.55%
TimesFM-1.0	0.926	0.876	+5.39%	0.917	0.881	+3.88%
VisionTS++	0.928	0.888	+4.34%	0.952	0.907	+4.77%
Chronos-Bolt	0.955	0.881	+7.75%	0.957	0.886	+7.41%
Moirai-1.1	1.074	0.958	+10.80%	1.076	0.984	+8.55%
Geo. Mean	0.915	0.869	+4.95%	0.901	0.869	+3.59%

Table 16: Forecasting performance when PRE is trained in a single stage using a mixture of synthetic and real residuals. MASE is reported.

TSFM	In-domain			Zero-shot		
	Base	Ablated PRE	Gain	Base	Ablated PRE	Gain
Toto-2	0.857	0.837	+2.36%	0.823	0.821	+0.25%
TimesFM-2.5	0.853	0.838	+1.74%	0.834	0.828	+0.78%
TiREx	0.866	0.842	+2.69%	0.841	0.828	+1.48%
TimesFM-2.0	0.891	0.853	+4.23%	0.880	0.858	+2.49%
Moirai-2	0.904	0.855	+5.43%	0.876	0.847	+3.37%
Chronos-2	0.914	0.849	+7.17%	0.886	0.845	+4.71%
TimesFM-1.0	0.926	0.876	+5.44%	0.917	0.874	+4.68%
VisionTS++	0.928	0.876	+5.63%	0.952	0.897	+5.80%
Chronos-Bolt	0.955	0.873	+8.52%	0.957	0.882	+7.86%
Moirai-1.1	1.074	0.941	+12.37%	1.076	0.969	+9.96%
Geo. Mean	0.915	0.863	+5.61%	0.901	0.864	+4.18%

Table 17: Forecasting performance of 10 TSFMs with a forecast memory size of $K = 1$.

TSFM	In-domain			Zero-shot		
	Base	PRE ($K = 1$)	Gain	Base	PRE ($K = 1$)	Gain
Toto-2	0.857	0.840	+2.02%	0.823	0.816	+0.79%
TimesFM-2.5	0.853	0.840	+1.50%	0.834	0.827	+0.90%
TiREx	0.866	0.849	+1.89%	0.841	0.830	+1.30%
TimesFM-2.0	0.891	0.864	+2.99%	0.880	0.864	+1.81%
Moirai-2	0.904	0.860	+4.86%	0.876	0.857	+2.18%
Chronos-2	0.914	0.857	+6.28%	0.886	0.848	+4.38%
TimesFM-1.0	0.926	0.894	+3.45%	0.917	0.896	+2.29%
VisionTS++	0.928	0.897	+3.42%	0.952	0.915	+3.84%
Chronos-Bolt	0.955	0.893	+6.42%	0.957	0.902	+5.74%
Moirai-1.1	1.074	1.010	+5.90%	1.076	1.023	+4.92%
Geo. Mean	0.915	0.879	+3.89%	0.901	0.876	+2.83%

Table 22 reports the complete inference-time and memory results across different forecast horizons. At $H = 96$, the median additional inference times are 32.51 ms, 16.68 ms, and 6.94 ms with one CPU thread, eight CPU threads, and GPU execution, respectively. The corresponding p95 times are 33.88 ms, 16.88 ms, and 6.99 ms, remaining close to the median values. As the forecast horizon increases, the inference latency grows more noticeably on CPUs, but remains modest: even at the longest horizon of $H = 864$, the median inference time with eight CPU threads is 95.08 ms.

Table 18: Forecasting performance of 10 TSFMs with a forecast memory size of $K = 4$.

TSFM	In-domain			Zero-shot		
	Base	PRE ($K = 4$)	Gain	Base	PRE ($K = 4$)	Gain
Toto-2	0.857	0.839	+2.14%	0.823	0.814	+1.08%
TimesFM-2.5	0.853	0.837	+1.85%	0.834	0.826	+0.98%
TiREx	0.866	0.847	+2.21%	0.841	0.828	+1.48%
TimesFM-2.0	0.891	0.862	+3.30%	0.880	0.862	+2.00%
Moirai-2	0.904	0.856	+5.29%	0.876	0.854	+2.50%
Chronos-2	0.914	0.854	+6.60%	0.886	0.846	+4.55%
TimesFM-1.0	0.926	0.891	+3.78%	0.917	0.890	+2.89%
VisionTS++	0.928	0.896	+3.51%	0.952	0.916	+3.75%
Chronos-Bolt	0.955	0.887	+7.06%	0.957	0.894	+6.67%
Moirai-1.1	1.074	0.989	+7.85%	1.076	1.014	+5.79%
Geo. Mean	0.915	0.875	+4.38%	0.901	0.873	+3.19%

Table 19: Forecasting performance of 10 TSFMs with a forecast memory size of $K = 8$.

TSFM	In-domain			Zero-shot		
	Base	PRE ($K = 8$)	Gain	Base	PRE ($K = 8$)	Gain
Toto-2	0.857	0.829	+3.35%	0.823	0.812	+1.28%
TimesFM-2.5	0.853	0.833	+2.27%	0.834	0.822	+1.48%
TiREx	0.866	0.838	+3.15%	0.841	0.826	+1.78%
TimesFM-2.0	0.891	0.853	+4.22%	0.880	0.855	+2.77%
Moirai-2	0.904	0.845	+6.48%	0.876	0.851	+2.84%
Chronos-2	0.914	0.843	+7.74%	0.886	0.846	+4.59%
TimesFM-1.0	0.926	0.878	+5.23%	0.917	0.883	+3.70%
VisionTS++	0.928	0.881	+5.12%	0.952	0.897	+5.81%
Chronos-Bolt	0.955	0.872	+8.61%	0.957	0.888	+7.28%
Moirai-1.1	1.074	0.972	+9.43%	1.076	1.001	+6.99%
Geo. Mean	0.915	0.864	+5.59%	0.901	0.867	+3.88%

Table 20: Forecasting performance of 10 TSFMs with a forecast memory size of $K = 32$.

TSFM	In-domain			Zero-shot		
	Base	PRE ($K = 32$)	Gain	Base	PRE ($K = 32$)	Gain
Toto-2	0.857	0.821	+4.26%	0.823	0.804	+2.24%
TimesFM-2.5	0.853	0.826	+3.09%	0.834	0.816	+2.20%
TiREx	0.866	0.827	+4.43%	0.841	0.816	+3.00%
TimesFM-2.0	0.891	0.847	+4.97%	0.880	0.849	+3.54%
Moirai-2	0.904	0.839	+7.12%	0.876	0.836	+4.57%
Chronos-2	0.914	0.837	+8.47%	0.886	0.831	+6.21%
TimesFM-1.0	0.926	0.869	+6.11%	0.917	0.875	+4.50%
VisionTS++	0.928	0.875	+5.76%	0.952	0.891	+6.42%
Chronos-Bolt	0.955	0.858	+10.15%	0.957	0.866	+9.55%
Moirai-1.1	1.074	0.950	+11.53%	1.076	0.984	+8.59%
Geo. Mean	0.915	0.854	+6.63%	0.901	0.855	+5.11%

The GPU latency increases more gently and remains only 9.27 ms at $H = 864$. Meanwhile, the additional memory usage remains low across all evaluated settings. These results demonstrate that PRE maintains low deployment overhead across different hardware settings and forecast horizons. This low overhead makes PRE practical for online deployment.

Table 21: Forecasting performance of 10 TSFMs with a forecast memory size of $K = 64$.

TSFM	In-domain			Zero-shot		
	Base	PRE ($K = 64$)	Gain	Base	PRE ($K = 64$)	Gain
Toto-2	0.857	0.825	+3.73%	0.823	0.806	+2.02%
TimesFM-2.5	0.853	0.830	+2.63%	0.834	0.814	+2.44%
TiREx	0.866	0.828	+4.31%	0.841	0.815	+3.10%
TimesFM-2.0	0.891	0.848	+4.81%	0.880	0.849	+3.54%
Moirai-2	0.904	0.839	+7.20%	0.876	0.833	+4.97%
Chronos-2	0.914	0.836	+8.54%	0.886	0.832	+6.10%
TimesFM-1.0	0.926	0.871	+5.94%	0.917	0.869	+5.17%
VisionTS++	0.928	0.875	+5.78%	0.952	0.891	+6.43%
Chronos-Bolt	0.955	0.859	+9.98%	0.957	0.866	+9.53%
Moirai-1.1	1.074	0.949	+11.59%	1.076	0.980	+8.91%
Geo. Mean	0.915	0.855	+6.49%	0.901	0.854	+5.25%

Table 22: Detailed online deployment cost of PRE. We report the median and 95th-percentile (p95) additional inference time per forecast, and the peak additional memory usage.

Deployment	Metric	Forecast Horizon H				
		24	**96**	**336**	**672**	**864**
CPU (1 thread)	Median Time (ms)	17.20	32.51	92.02	203.53	332.50
	p95 Time (ms)	19.76	33.88	94.53	205.87	333.85
	Memory (MiB)	100.34	100.34	100.34	100.34	100.34
CPU (8 threads)	Median Time (ms)	11.81	16.68	36.02	67.89	95.08
	p95 Time (ms)	12.90	16.88	36.31	68.75	95.74
	Memory (MiB)	94.89	94.89	94.89	94.89	94.89
GPU	Median Time (ms)	6.89	6.94	9.22	9.23	9.27
	p95 Time (ms)	6.93	6.99	9.26	9.27	9.32
	Memory (MiB)	69.87	70.89	76.16	95.68	110.25

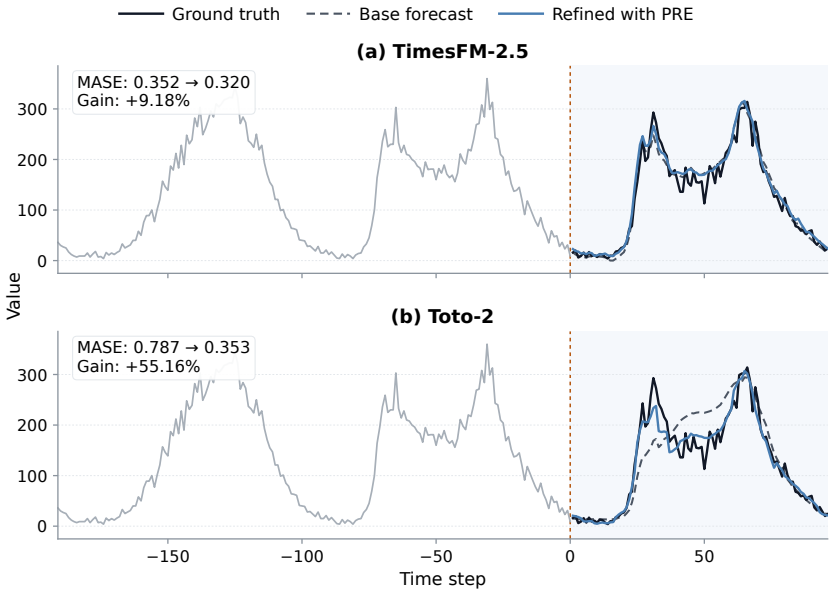

Figure 4: Visualization of PRE's forecast correction on `Finland_Traffic (15T)` under in-domain evaluation. (a) and (b) show results for TimesFM-2.5 and Toto-2, respectively, on the same forecasting window.

D.6 Visualization of Forecast Correction

To provide an intuitive illustration of PRE's forecast correction, this section presents case studies on two datasets. Figures 4 and 5 present forecasting examples on `Finland_Traffic (15T)` and `current_velocity (10T)` under the in-domain and zero-shot settings, respectively. Each figure shows the base forecasts from two leading TSFMs, TimesFM-2.5 and Toto-2, alongside the forecasts refined with PRE. The reported MASE values and gains refer to the displayed forecasting windows.

As shown in Figure 4, TimesFM-2.5 already provides an accurate base forecast in this example. PRE makes small adjustments that bring the forecast closer to the peaks and troughs of the ground truth, yielding a relative MASE reduction of 9.18%. For Toto-2, although the base forecast captures the overall trend, it is overly smooth over forecast steps 20–70, underestimating the first peak and overestimating the values between two peaks. PRE alleviates this oversmoothing, leading to a substantial relative MASE reduction of 55.16%. As shown in Figure 5, both base forecasts exhibit temporal lag to the ground truth. For TimesFM-2.5, the lag is relatively small, and PRE brings the

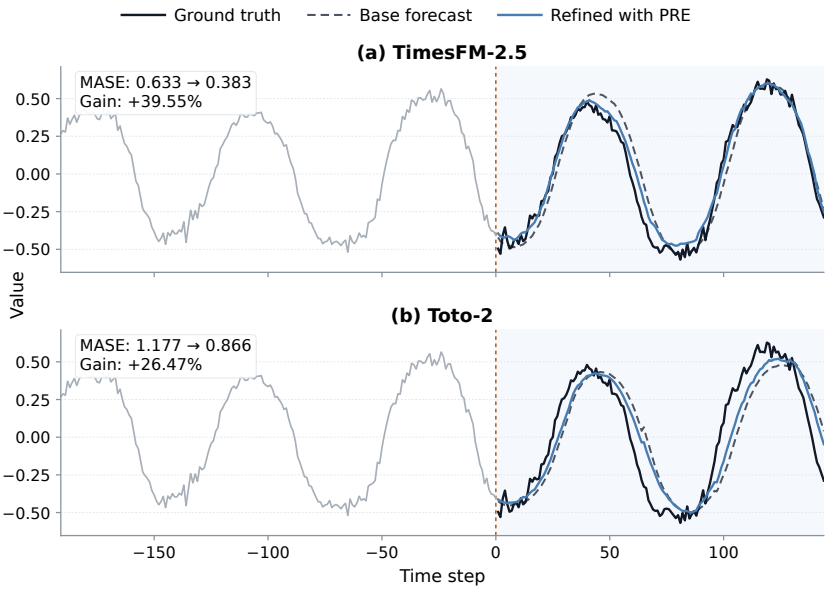

Figure 5: Visualization of PRE's forecast correction on `current_velocity (10T)` under zero-shot evaluation. (a) and (b) show results for TimesFM-2.5 and Toto-2, respectively, on the same forecasting window.

forecast into closer temporal alignment with the observed trajectory. For Toto-2, the initial lag is more pronounced, but PRE noticeably reduces it.