# ESTIMATE WHAT REPEATS, LEARN WHAT REMAINS: POST-HOC CORRECTION OF FROZEN TIME SERIES FORECASTERS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Deep forecasters and pretrained foundation models have made time-series forecasting far more accurate, yet the errors of a trained model are not unstructured noise. Whether a forecaster overpredicts or underpredicts often depends on where the target falls within the daily cycle and on how many steps ahead the forecast is made, and the same dependence is already present on the training data. Existing post-hoc correctors lower the average error but leave much of this recurring pattern in place. We introduce LIRA (Lookup and Input-conditioned Residual Adjustment), a correction framework for frozen forecasters that first estimates what repeats and then learns what remains. A recurring error lookup pools historical residuals by target cycle position and forecast lead, and a channel-shared linear head is trained on the residual left after the lookup. Because the lookup stores statistics rather than learned weights, newly observed forecast errors update the lookup alone, so online correction needs neither additional backbone calls nor gradient optimization. Across five task-specific backbones and four foundation models, online LIRA lowers horizon-averaged MSE in every backbone–dataset pair, by 8.77% and 17.64% on average, and matches or outperforms the post-hoc correctors $\delta$-Adapter and PIR in most task-specific pairs, with lower correction computation and storage costs.

## 1 INTRODUCTION

Deep forecasters and pretrained foundation models have raised time-series forecasting accuracy considerably (Nie et al., 2023; Liu et al., 2024; Wang et al., 2024a; Das et al., 2023b; Ansari et al., 2024). The errors of a trained forecaster, however, are not unstructured noise. Whether a forecaster overpredicts or underpredicts often depends on where the target falls within the daily cycle and on the forecast lead, and the same dependence is already visible on the training data. We estimate that recurring error from historical residuals, subtract the estimate, and learn only the error that remains, all without touching the forecaster.

Correcting forecasts has a long history in statistical post-processing (Glahn & Lowry, 1972). Recent correctors learn bounded adjustments (Liang et al., 2026), revise forecasts with local context and retrieved instances (Liu et al., 2026), add causality-inspired safeguards (Xie et al., 2025), or update the correction online (Im & Kwon, 2026), and residual modeling also appears in hybrid and cycle-based forecasting (Zhang, 2003; Lin et al., 2024). These methods establish that residual structure can improve forecasting, but a lower average error does not show whether recurring cycle–lead patterns have been removed. CRC already separates linear correction from nonlinear refinement (Xie et al., 2025); our distinction is to first pool historical residuals at matched target cycle positions and forecast leads, then learn the remainder.

Estimating recurring error requires deciding which residuals belong together. Each multi-step forecast yields a residual window, and windows from consecutive issue times overlap, so one target timestamp is predicted several times at different leads. We therefore index every residual by two coordinates. The *target cycle position* is where the predicted time step falls within a cycle such as the day, and the *forecast lead* is the number of steps ahead. Forecasts that share both coordinates



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.

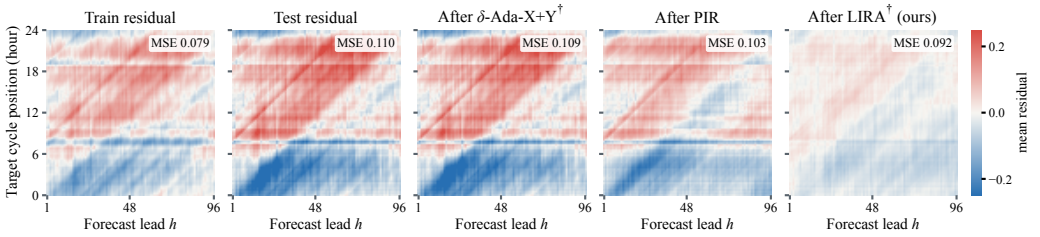



Figure 1: Recurring error before and after correction. Comparison of PatchTST residuals on the training and test splits, followed by test residuals after $\delta$-Ada-X+Y$^\dagger$, PIR, and LIRA$^\dagger$, for one ETTm2 channel ($H = 96$). Each cell shows the mean signed residual at one target cycle position (row) and forecast lead (column). All panels share one color scale. † denotes online correction.

face the same forecasting condition, so pooling the signed residuals over each cycle–lead pair gives a per-channel map of where the backbone systematically overpredicts or underpredicts.

Figure 1 shows this map for PatchTST on one ETTm2 channel. The backbone overpredicts in the early morning and underpredicts from late morning through the evening, and the bands estimated on the training split reappear on the test split. The pattern is therefore not a symptom of distribution shift alone, because the backbone minimized the training loss on exactly the data where the bands first appear. Both $\delta$-Adapter (Liang et al., 2026) and PIR (Liu et al., 2026) lower the MSE on this channel, but the bands remain clearly visible after either correction. Estimating the recurring pattern directly and subtracting the estimate removes most of the bands (last panel).

To this end, we propose LIRA (*Lookup and Input-conditioned Residual Adjustment*), which corrects a frozen forecaster in two stages. A *Recurring Error Lookup* first estimates the recurring error by pooling historical backbone residuals over target cycle position and forecast lead, and calibrated weights control how much the cycle, lead, and interaction patterns contribute. Two forecasts with the same cycle position and lead may still need different corrections because the recent observations differ, so a *Remainder Prediction Head* then takes the recent observations and the original forecast and predicts the residual left after the lookup correction. The head is a single linear map shared across channels and is trained on out-of-fold lookup residuals, so the lookup estimate used for each training target never contains that target itself.

Because the lookup stores pooled statistics rather than learned parameters, LIRA adapts online without optimization. Once the full target window of a forecast has been observed, the resulting residual updates the statistics while the backbone, the head, and the calibration coefficients stay fixed, and waiting for the full window rules out any use of future labels. Fitting needs only historical forecasts and observations, and the same holds for a pretrained foundation backbone. LIRA never reads the parameters, gradients, or hidden states of the backbone, so the method applies unchanged to a model reachable only through the forecasts.

We evaluate LIRA on five task-specific backbones across seven datasets and on four pretrained foundation models across six datasets. Online LIRA lowers MSE in every backbone–dataset pair, by 8.77% and 17.64% on average, and is competitive with $\delta$-Adapter, PIR, and online test-time adaptation methods at lower correction cost.

Our contributions are:

- **Analysis of recurring forecast errors.** We show that residuals of trained forecasters carry a stable pattern over target cycle position and forecast lead, already present on the training split, and that a calibrated lookup of the pattern alone improves accuracy.
- **Lookup correction followed by remainder learning.** We introduce LIRA, which pairs an explicit recurring-error lookup with an input-dependent channel-shared linear head trained on the residual left after the lookup.
- **Online correction through statistical updates alone.** We refresh residual statistics from fully observed targets while the backbone, head, and calibration coefficients stay fixed, so adaptation needs neither gradient updates nor backbone calls.

## 2 Related Work

**Forecasting architectures and foundation models.** PatchTST processes temporal patches with channel-shared weights, and iTransformer treats variables as tokens (Nie et al., 2023; Liu et al., 2024). DLinear, TimeMixer, and TimeBase forecast from decomposition, multiscale mixing, and temporal bases (Zeng et al., 2023; Wang et al., 2024a; Huang et al., 2025), and CycleNet learns the recurring cycle of the series inside the model and forecasts the residual after removing that cycle (Lin et al., 2024). TimesFM, Chronos, Chronos-2, Sundial, and Toto-2.0 are pretrained on large datasets (Das et al., 2023b; Ansari et al., 2024; 2025; Liu et al., 2025; Khwaja et al., 2026).

**Post-processing and residual correction.** Model Output Statistics estimates statistical relationships between numerical model forecasts and observed weather (Glahn & Lowry, 1972), and hybrid ARIMA–neural forecasting fits a neural network to the residuals of an ARIMA model (Zhang, 2003). Recent methods extend post-processing and residual modeling to neural forecasts. $\delta$-Adapter learns bounded input and output adjustments while keeping the backbone frozen (Liang et al., 2026). PIR revises forecasts with local context and retrieved historical instances and is trained jointly with the forecasting model (Liu et al., 2026). CRC combines a linear ridge correction with nonlinear residual refinement, using causality-inspired representations and safeguards against harmful corrections (Xie et al., 2025). LIRA estimates the recurring component by pooling residuals at matched cycle positions and forecast leads, then fits a head to the remainder.

**Online correction and adaptation.** TAFAS attaches gated calibration modules to the input and the output of a frozen forecaster and updates the modules by gradient descent from partially observed targets (Kim et al., 2025). PETSA replaces the modules with low-rank adapters and a structural loss (Medeiros et al., 2025), and DynaTTA conditions the gate and the update rate on an estimate of distribution shift (Grover & Etemad, 2025). COSA uses an output-only module with a gated linear residual conditioned on statistics of recently observed targets (Im & Kwon, 2026). All four methods adapt learned parameters by gradient descent, and the first three also need forward passes and gradients through the backbone. LIRA updates only empirical residual statistics and only after a full target window has been observed, while the head and the calibration coefficients stay fixed.

## 3 LIRA: Lookup and Input-conditioned Residual Adjustment

LIRA corrects a frozen backbone forecast through two components: a *Recurring Error Lookup*, which estimates recurring error patterns from historical residuals, and a *Remainder Prediction Head*, which predicts the remaining error from the current input. We refer to them as the error lookup and remainder head below. Figure 2 provides an overview of LIRA.

Given a history window $X_t \in \mathbb{R}^{L\times C}$ containing the latest $L$ observations of $C$ channels at time $t$, our goal is to predict the next $H$ observations, denoted by $Y_t \in \mathbb{R}^{H\times C}$. The frozen backbone forecaster $\mathcal{F}$ produces the uncorrected forecast $\widehat{Y}_t = \mathcal{F}(X_t) \in \mathbb{R}^{H\times C}$. LIRA adds the outputs of its two components to obtain the corrected forecast $\widetilde{Y}_t \in \mathbb{R}^{H\times C}$:

$$\widetilde{Y}_t = \widehat{Y}_t + \Delta_t^{\text{lookup}} + \Delta_t^{\text{head}}. \tag{1}$$

$\Delta_t^{\text{lookup}} \in \mathbb{R}^{H\times C}$ is the correction produced by the error lookup from recurring patterns in historical forecast errors, and $\Delta_t^{\text{head}} \in \mathbb{R}^{H\times C}$ is the input-conditioned correction from the remainder head.

### 3.1 Target Cycle Position and Forecast Lead

We organize historical forecast errors using two coordinates: *target cycle position*, where the predicted time step falls within a cycle, and *forecast lead*, how many steps ahead the prediction is made. For example, under a daily cycle, forecasts for 21:00 on different days share the same target cycle position, while forecasts made one and three hours before 21:00 have different leads.

Let $t$ denote the last observed time step serving as the forecast origin, $S$ the cycle length in time steps, and $\tau_t \in \{0, \dots, S-1\}$ the cycle position at $t$. For a forecast made $h \in \{1, \dots, H\}$ steps ahead, the target time $t+h$ has cycle position

$$p_{t,h} = (\tau_t + h) \bmod S. \tag{2}$$

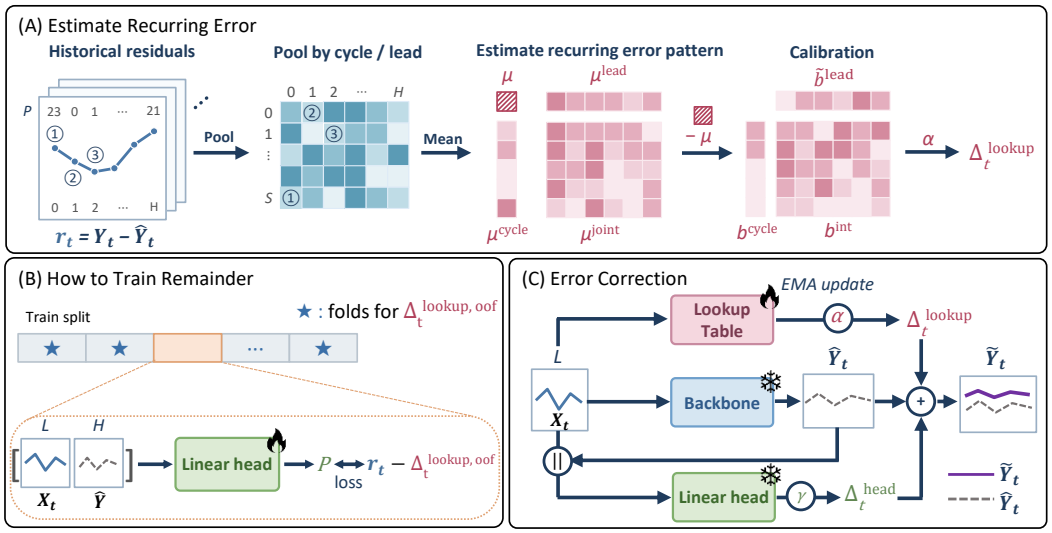



Figure 2: Overall mechanism of LIRA. (A) Estimating recurring errors by target cycle position and forecast lead, using one channel of hourly data as an example. (B) Training the head to predict the error left after an out-of-fold (OOF) lookup correction. (C) Adding lookup and head corrections to the frozen backbone forecast, and updating lookup statistics when full targets become available.

We use target cycle position and forecast lead to organize historical residuals and compute the lookup correction $\Delta_t^{\text{lookup}}$, as described below.

### 3.2 ESTIMATING RECURRING ERROR PATTERNS

We estimate recurring error patterns from historical backbone residuals, defined as $r_t = Y_t - \widehat{Y}_t$. For each channel $j \in \{1, \dots, C\}$, these residuals are summarized by four empirical means: the global mean $\mu \in \mathbb{R}^C$, the cycle mean $\mu^{\text{cycle}} \in \mathbb{R}^{S \times C}$, the lead mean $\mu^{\text{lead}} \in \mathbb{R}^{H \times C}$, and the joint mean $\mu^{\text{joint}} \in \mathbb{R}^{S \times H \times C}$. These means are computed as

$$\begin{aligned} \mu_j &= \frac{1}{NH} \sum_t \sum_{h=1}^{H} r_{t,h,j}, \quad \mu^{\text{cycle}}_{p,j} = \frac{1}{n_p} \sum_t \sum_{h=1}^{H} \mathbf{1}\{p_{t,h} = p\}\, r_{t,h,j}, \\ \mu^{\text{lead}}_{h,j} &= \frac{1}{N} \sum_t r_{t,h,j}, \quad \mu^{\text{joint}}_{p,h,j} = \frac{1}{n_{p,h}} \sum_t \mathbf{1}\{p_{t,h} = p\}\, r_{t,h,j}, \end{aligned} \tag{3}$$

where $N$ denotes the number of residual windows from the training split, and all sums over $t$ run over these windows. The entry $r_{t,h,j}$ is the residual for the forecast at origin $t$, lead $h$, and channel $j$. For each channel, $n_p$ and $n_{p,h}$ count residual entries at cycle position $p$ and at the pair $(p, h)$, respectively. The indicator $\mathbf{1}\{\cdot\}$ equals one when its condition holds and zero otherwise.

To capture cycle- and lead-dependent deviations from the global mean, we subtract it from the corresponding means to obtain the cycle pattern $b^{\text{cycle}}$ and lead pattern $b^{\text{lead}}$. Additionally, because some error patterns may be specific to combinations of cycle position and forecast lead, we introduce an interaction pattern $b^{\text{int}}$ to capture them, with all three patterns defined as follows:

$$\begin{aligned} b^{\text{cycle}}_{p,j} &= \mu^{\text{cycle}}_{p,j} - \mu_j, \\ b^{\text{lead}}_{h,j} &= \mu^{\text{lead}}_{h,j} - \mu_j, \\ b^{\text{int}}_{p,h,j} &= \mu^{\text{joint}}_{p,h,j} - \mu_j - b^{\text{cycle}}_{p,j} - b^{\text{lead}}_{h,j}. \end{aligned} \tag{4}$$

### 3.3 Calibrating the Lookup Correction

The cycle, lead, and interaction patterns may differ in how well they transfer to future forecasts. We therefore combine them using coefficients fitted on validation data. We divide the forecast horizon into $Q$ consecutive groups and denote the group containing lead $h$ by $q(h)$. For each pattern, we fit a separate coefficient for each channel and forecast group.

Before calibration, we subtract each group's mean from the lead pattern so that its coefficient scales only within-group variation:

$$\widetilde{b}^{\text{lead}}_{h,j} = b^{\text{lead}}_{h,j} - \frac{1}{|\mathcal{H}_{q(h)}|} \sum_{k \in \mathcal{H}_{q(h)}} b^{\text{lead}}_{k,j}, \tag{5}$$

where $\mathcal{H}_q$ is the set of leads in group $q$. We compute the lookup correction by combining the patterns at the target cycle position and forecast lead:

$$\Delta^{\text{lookup}}_{t,h,j} = \alpha^{\text{cycle}}_{j,q(h)} b^{\text{cycle}}_{p_{t,h},j} + \alpha^{\text{lead}}_{j,q(h)} \widetilde{b}^{\text{lead}}_{h,j} + \alpha^{\text{int}}_{j,q(h)} b^{\text{int}}_{p_{t,h},h,j}. \tag{6}$$

We fit the lookup calibration coefficients on validation data to minimize the squared error between $\widehat{Y}_t + \Delta^{\text{lookup}}_t$ and $Y_t$. These coefficients are fixed before training the remainder head.

### 3.4 Remainder Prediction Head

Forecasts sharing the same lookup correction can still have different errors depending on the current observations. We therefore introduce an input-conditioned remainder head to predict the error left after lookup correction. We implement this head as a lightweight linear layer shared across channels. For each channel $j$, we concatenate the observed history and the original backbone forecast:

$$z_{t,j} = \begin{bmatrix} X_{t,:,j} \\ \widehat{Y}_{t,:,j} \end{bmatrix} \in \mathbb{R}^{L+H}, \qquad P_{t,:,j} = W z_{t,j} + \beta. \tag{7}$$

where $W \in \mathbb{R}^{H \times (L+H)}$ and $\beta \in \mathbb{R}^H$, and $P_t \in \mathbb{R}^{H \times C}$ is the predicted remainder before calibration.

We train the remainder head on the difference between the backbone residual and the lookup correction. We compute this correction out of fold (OOF), excluding each example's own residual from the lookup statistics. Training windows are divided into contiguous folds; each fold's residual statistics are estimated from the other folds, excluding windows with overlapping targets. Using these statistics and the fixed lookup calibration coefficients, we compute the OOF lookup correction $\Delta^{\text{lookup,oof}}_t$ via Eq. (6). The head's training target is $r_t - \Delta^{\text{lookup,oof}}_t$.

As in lookup calibration, we fit a coefficient $\gamma_{j,q(h)}$ for each channel and forecast group to control the strength of the head's correction:

$$\Delta^{\text{head}}_{t,h,j} = \gamma_{j,q(h)} P_{t,h,j}. \tag{8}$$

With the head parameters fixed, we fit these coefficients on validation data to minimize the squared error between $\widehat{Y}_t + \Delta^{\text{lookup}}_t + \Delta^{\text{head}}_t$ and $Y_t$.

### 3.5 Online Updates of Residual Statistics

After deployment, recurring error patterns may change, reducing the accuracy of lookup estimates derived from historical residuals. We update the lookup statistics using an exponential moving average (EMA), allowing recent residuals to adjust historical estimates of recurring errors.

For a forecast issued at time $t$, the complete residual

$$r_t = Y_t - \widehat{Y}_t \tag{9}$$

becomes available at time $t+H$, when its entire target window has been observed. To prevent future-label leakage, only forecasts with fully observed targets contribute to the updates. Their residuals update the global, cycle, lead, and joint statistics used to construct subsequent lookup corrections. The backbone, remainder head, and calibration coefficients remain fixed, allowing adaptation of recurring error estimates without further model training.

Table 1: **Task-specific forecasting performance.** MSE/MAE averaged over four forecast horizons. Best scores are bold; second-best scores are underlined.

| | | | Baselines | | | | Ours | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Model | Backbone | $\delta$-Ada-Y | $\delta$-Ada-Y$^\dagger$ | $\delta$-Ada-X+Y$^\dagger$ | PIR | LIRA | LIRA$^\dagger$ |
| ETTh1 | PatchTST | 0.451/0.450 | 0.451/0.450 | 0.451/0.450 | 0.451/0.450 | 0.447/0.450 | 0.440/0.445 | **0.439/0.444** |
| | iTransformer | 0.464/0.455 | 0.464/0.455 | 0.464/0.455 | 0.464/0.455 | **0.434/0.439** | 0.443/0.444 | 0.441/0.443 |
| | DLinear | 0.460/0.457 | 0.461/0.457 | 0.460/0.456 | 0.461/0.457 | 0.450/0.454 | 0.437/0.440 | **0.436/0.439** |
| | TimeMixer | 0.467/0.448 | 0.465/0.448 | 0.466/0.448 | 0.463/0.447 | **0.444**/0.440 | 0.446/0.437 | 0.445/**0.436** |
| | TimeBase | 0.463/0.429 | 0.461/0.428 | 0.461/0.429 | 0.458/0.428 | **0.436/0.425** | 0.441/0.429 | 0.438/0.428 |
| ETTh2 | PatchTST | 0.385/0.413 | 0.382/0.411 | 0.384/0.412 | 0.379/0.409 | 0.391/0.415 | **0.365/0.398** | **0.365/0.398** |
| | iTransformer | 0.383/0.407 | 0.381/0.406 | 0.384/0.407 | 0.381/0.406 | 0.379/0.404 | **0.364**/0.394 | 0.365/**0.393** |
| | DLinear | 0.564/0.519 | 0.566/0.520 | 0.558/0.516 | 0.560/0.517 | 0.545/0.509 | **0.466/0.464** | 0.467/0.465 |
| | TimeMixer | 0.395/0.411 | 0.393/0.410 | 0.394/0.411 | 0.390/0.409 | 0.389/0.408 | **0.373/0.399** | 0.374/**0.399** |
| | TimeBase | 0.405/0.421 | 0.403/0.419 | 0.406/0.421 | 0.400/0.418 | 0.387/0.408 | **0.382/0.405** | 0.383/**0.405** |
| ETTm1 | PatchTST | 0.389/0.403 | 0.388/0.402 | 0.389/0.403 | 0.387/0.402 | 0.381/0.397 | 0.369/0.397 | **0.359/0.383** |
| | iTransformer | 0.407/0.411 | 0.406/0.411 | 0.407/0.411 | 0.405/0.411 | 0.386/0.398 | 0.385/0.401 | **0.373/0.388** |
| | DLinear | 0.404/0.408 | 0.404/0.408 | 0.403/0.407 | 0.401/0.405 | 0.377/0.394 | 0.365/0.387 | **0.360/0.382** |
| | TimeMixer | 0.383/0.398 | 0.383/0.398 | 0.383/0.398 | 0.382/0.397 | 0.374/0.393 | 0.370/0.392 | **0.360/0.381** |
| | TimeBase | 0.429/0.419 | 0.428/0.418 | 0.429/0.419 | 0.427/0.417 | 0.381/0.392 | 0.372/0.392 | **0.359/0.382** |
| ETTm2 | PatchTST | 0.290/0.334 | 0.289/0.335 | 0.292/0.336 | 0.287/0.333 | 0.296/0.340 | 0.275/0.323 | **0.273/0.322** |
| | iTransformer | 0.291/0.334 | 0.289/0.335 | 0.292/0.336 | 0.290/0.336 | 0.289/0.335 | 0.272/0.320 | **0.271/0.318** |
| | DLinear | 0.355/0.402 | 0.356/0.402 | 0.350/0.397 | 0.345/0.393 | 0.368/0.405 | **0.287/0.344** | **0.287/0.344** |
| | TimeMixer | 0.277/0.324 | 0.275/0.324 | 0.277/0.325 | 0.274/0.323 | 0.278/0.326 | 0.261/0.314 | **0.260/0.312** |
| | TimeBase | 0.290/0.332 | 0.288/0.332 | 0.291/0.333 | 0.288/0.332 | 0.284/0.330 | 0.269/0.316 | **0.268/0.314** |
| Weather | PatchTST | 0.258/0.280 | 0.247/0.278 | 0.256/0.287 | 0.244/0.278 | 0.254/0.278 | 0.236/0.279 | **0.231/0.268** |
| | iTransformer | 0.260/0.281 | 0.248/0.278 | 0.258/0.287 | 0.245/0.277 | 0.258/0.280 | 0.239/0.277 | **0.235/0.270** |
| | DLinear | 0.265/0.317 | 0.256/0.306 | 0.260/0.307 | **0.247/0.293** | 0.251/0.300 | 0.260/0.306 | 0.250/0.294 |
| | TimeMixer | 0.244/0.273 | 0.237/0.273 | 0.244/0.281 | 0.235/0.275 | 0.245/0.275 | 0.228/0.272 | **0.222/0.263** |
| | TimeBase | 0.252/0.279 | 0.242/0.277 | 0.251/0.287 | 0.242/0.283 | 0.251/0.287 | 0.234/0.282 | **0.227/0.270** |
| ECL | PatchTST | 0.204/0.294 | 0.195/0.282 | 0.201/0.287 | **0.188**/0.278 | 0.198/**0.276** | 0.192/0.278 | 0.192/0.278 |
| | iTransformer | 0.180/0.270 | 0.174/0.267 | 0.181/0.274 | **0.171/0.265** | 0.176/**0.265** | 0.175/0.267 | 0.175/0.267 |
| | DLinear | 0.225/0.319 | 0.214/0.308 | 0.221/0.313 | 0.205/0.300 | **0.195/0.281** | 0.209/0.297 | 0.208/0.295 |
| | TimeMixer | 0.185/0.275 | 0.180/0.271 | 0.186/0.277 | **0.176/0.268** | 0.182/0.272 | 0.182/0.272 | 0.182/0.272 |
| | TimeBase | 0.227/0.295 | 0.216/0.292 | 0.226/0.301 | 0.208/0.288 | **0.199/0.280** | 0.211/0.290 | 0.211/0.290 |
| Solar | PatchTST | 0.269/0.307 | 0.260/0.293 | 0.267/0.299 | 0.233/0.278 | 0.246/0.288 | 0.241/0.281 | **0.226/0.270** |
| | iTransformer | 0.241/0.271 | 0.235/0.269 | 0.243/0.284 | **0.218**/0.266 | 0.239/0.272 | 0.228/0.273 | **0.218/0.260** |
| | DLinear | 0.330/0.401 | 0.309/0.382 | 0.321/0.392 | 0.278/0.356 | **0.261/0.324** | 0.287/0.347 | 0.276/0.333 |
| | TimeMixer | 0.218/0.289 | 0.218/0.288 | 0.222/0.296 | 0.217/0.287 | 0.219/0.290 | 0.216/0.287 | **0.201/0.272** |
| | TimeBase | 0.385/0.375 | 0.360/0.361 | 0.370/0.370 | 0.311/0.329 | 0.273/0.323 | 0.270/0.326 | **0.266/0.320** |

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

We evaluate LIRA on seven benchmark datasets with $L = 96$ and forecast horizons $H \in \{96, 192, 336, 720\}$, using diverse task-specific forecasting backbones and pretrained foundation models. All experiments use a daily cycle, with $S$ set by the sampling interval ($S = 24$ for hourly data). We compare with the post-hoc correctors $\delta$-Adapter and PIR, and with the online test-time adaptation methods TAFAS, PETSA, DynaTTA, and COSA-P. Backbone fine-tuning is evaluated in Appendix C.11. $\delta$-Ada-Y adapts the output only, and $\delta$-Ada-X+Y adapts both the input and the output. Appendix C.2 describes the backbones and Appendix C.4 gives the online-adaptation protocol. Tables report MSE/MAE averaged over four horizons and three seeds. For LIRA and $\delta$-Adapter, a dagger (†) distinguishes online from offline variants.

## 4.2 OVERALL FORECASTING PERFORMANCE

**Post-hoc correctors on task-specific backbones.** Across the 35 horizon-averaged backbone–dataset pairs in Table 1, LIRA lowers MSE by $7.34\%$ offline and $8.77\%$ online on average, and MAE by $3.90\%$ and $5.43\%$, respectively. Online LIRA improves both metrics in every pair, with MSE reductions of $2.0\%$–$31.1\%$, and has lower MSE than $\delta$-Ada-X+Y$^\dagger$ and PIR in most pairs.

On ECL, offline and online LIRA improve the backbone, but less than the best competing methods. ECL has many channels, and using information across channels can help forecasting (Liu et al., 2024). $\delta$-Adapter and PIR use cross-channel information to correct forecasts (Liang et al., 2026; Liu et al., 2026). Our head instead applies one shared linear layer to each channel separately. The

Table 2: **Foundation-model forecasting performance.** MSE/MAE averaged over four forecast horizons. For each backbone and dataset, best scores are bold, including ties.

| Backbone | Correction | ETTh1 | ETTh2 | ETTm1 | ETTm2 | Weather | ECL |
|---|---|---|---|---|---|---|---|
| Chronos-2 | None | 0.492/0.439 | 0.406/0.405 | 0.616/0.477 | 0.318/0.338 | 0.307/0.291 | 0.234/0.294 |
| | $\delta$-Ada-Y$^\dagger$ | 0.490/0.438 | 0.407/0.407 | 0.614/0.477 | 0.319/0.341 | 0.293/0.296 | 0.230/0.297 |
| | LIRA$^\dagger$ | **0.451/0.435** | **0.388/0.397** | **0.375/0.397** | **0.274/0.316** | **0.244/0.283** | **0.205/0.283** |
| Sundial | None | 0.473/0.435 | 0.399/0.410 | 0.538/0.457 | 0.304/0.340 | 0.285/0.302 | 0.204/0.284 |
| | $\delta$-Ada-Y$^\dagger$ | 0.471/**0.434** | 0.401/0.412 | 0.537/0.457 | 0.306/0.342 | 0.281/0.310 | 0.204/0.288 |
| | LIRA$^\dagger$ | **0.449/0.434** | **0.380/0.401** | **0.364/0.388** | **0.272/0.318** | **0.247/0.290** | **0.192/0.277** |
| TimesFM-2.5 | None | 0.484/**0.435** | 0.413/0.406 | 0.875/0.584 | 0.372/0.378 | 0.316/0.300 | 0.239/0.308 |
| | $\delta$-Ada-Y$^\dagger$ | 0.483/**0.435** | 0.415/0.408 | 0.870/0.582 | 0.372/0.380 | 0.296/0.301 | 0.235/0.309 |
| | LIRA$^\dagger$ | **0.444/0.435** | **0.391/0.400** | **0.409/0.424** | **0.276/0.323** | **0.244/0.289** | **0.206/0.288** |
| Toto-2.0 | None | 0.515/0.447 | 0.409/0.406 | 0.834/0.571 | 0.332/0.355 | 0.316/0.305 | 0.245/0.316 |
| | $\delta$-Ada-Y$^\dagger$ | 0.513/0.447 | 0.409/0.408 | 0.830/0.570 | 0.332/0.357 | 0.299/0.305 | 0.238/0.313 |
| | LIRA$^\dagger$ | **0.455/0.442** | **0.386/0.396** | **0.406/0.422** | **0.270/0.314** | **0.251/0.293** | **0.204/0.287** |

channel-shared design may limit LIRA's gains on ECL. Even so, online LIRA lowers average MSE for every backbone on ECL while keeping the correction module small.

**Frozen foundation models.** Table 2 compares online LIRA with $\delta$-Ada-Y$^\dagger$ on four pretrained foundation models. The pretrained backbones remain frozen, while the correctors are fitted using task-specific historical targets. $\delta$-Ada-Y$^\dagger$ changes MSE by less than $2\%$ in 20 of the 24 backbone–dataset pairs. Online LIRA reduces MSE by $17.64\%$ on average and improves MSE in every pair. With online LIRA, the best foundation-model MSE is lower than the best uncorrected task-specific backbone in Table 1 on four of the six shared datasets and matches the best uncorrected backbone on Weather at the reported precision. For example, Sundial reaches 0.364 on ETTm1, compared with 0.383 for TimeMixer, without fine-tuning the pretrained backbone.

**Online test-time adaptation methods.** Table 3 compares LIRA with TAFAS (Kim et al., 2025), PETSA (Medeiros et al., 2025), DynaTTA (Grover & Etemad, 2025), and COSA-P (Im & Kwon, 2026), which adapt calibration modules around a frozen backbone through gradient steps on the test stream. The table covers ETTm1 and Weather with iTransformer and DLinear. Averaged over these four pairs, the adaptation methods reduce backbone MSE by at most $3.02\%$, while offline and online LIRA reduce backbone MSE by $6.26\%$ and $8.71\%$. Online LIRA has lower MSE and MAE than every adaptation method in every pair. Appendix C.4 gives the protocol and all 28 pairs.

Table 3: **Comparison with online test-time adaptation.** Horizon-averaged MSE/MAE. Best correction scores are bold; second-best scores are underlined. All 28 pairs are in Table 8.

| | | | Online test-time adaptation | | | | Ours | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Backbone | Backbone | TAFAS | PETSA | DynaTTA | COSA-P | LIRA | LIRA$^\dagger$ |
| ETTm1 | iTransformer | 0.407/0.411 | 0.391/0.410 | 0.393/0.411 | 0.395/0.408 | 0.394/0.409 | <u>0.385</u>/<u>0.401</u> | **0.373/0.388** |
| | DLinear | 0.404/0.408 | 0.392/0.400 | 0.391/0.400 | 0.396/0.402 | 0.393/0.402 | <u>0.365</u>/<u>0.387</u> | **0.360/0.382** |
| Weather | iTransformer | 0.260/0.281 | 0.250/0.282 | 0.260/0.286 | 0.257/0.291 | 0.253/0.279 | <u>0.239</u>/<u>0.277</u> | **0.235/0.270** |
| | DLinear | 0.265/0.317 | 0.262/0.313 | 0.268/0.309 | 0.265/0.317 | <u>0.257</u>/<u>0.302</u> | 0.260/0.306 | **0.250/0.294** |

**Correction cost.** Sharing the head across channels keeps the fitted scalar count nearly constant over the range in Figure 3b. Although lookup storage grows with $C$, LIRA requires fewer multiply–accumulates (MACs) and less core storage than the correctors at the plotted setting.

### 4.3 UNDERSTANDING LIRA

**RQ1: Do recurring patterns transfer?** Figure 1 already suggests that the pattern estimated on the training split reappears on the test split. To measure transfer, we fit and calibrate the lookup on earlier correction-training data and evaluate the frozen lookup on a later block (100 settings, Appendix D.1). Figure 4 compares cycle-only ($b^{\mathrm{cycle}}$), lead-only ($\widetilde{b}^{\mathrm{lead}}$), interaction-only ($b^{\mathrm{int}}$), and the combined lookup (full). The control labeled "full, shifted" reads the same full lookup with the cycle position shifted by half a day, keeping forecast leads fixed.

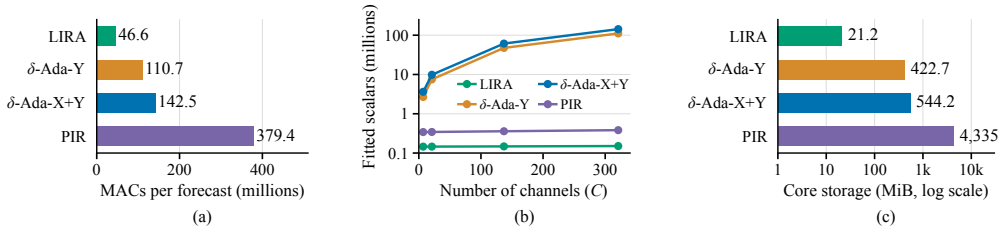



Figure 3: Correction cost. Comparison of LIRA with $\delta$-Adapter and PIR at $L = 96$ and $H = 336$. (a) MACs per forecast at $C = 321$. (b) Fitted weights, biases, and calibration coefficients versus channel count. (c) Core storage for fitted values, lookup statistics, and retrieval data at $C = 321$.

The full lookup improves most settings. The half-day shift removes most of the gain. The cycle-only lookup gives the largest standalone mean gain, while the lead-only lookup adds little. Combining all three components improves the mean gain over the cycle-only lookup. The gain therefore depends mainly on where the target falls within the day, with an additional contribution from cycle–lead interaction. Figure 5 illustrates the effect in a selected Weather test window at $H = 96$, where online LIRA recovers a peak missed by the backbone and competing correctors.

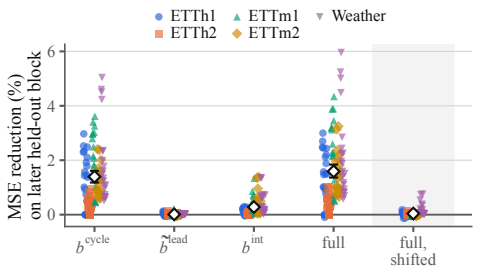



Figure 4: Lookup transfer to later data. Comparison of cycle, lead, interaction, and full lookups on data after the fitting period. "Full, shifted" shifts cycle positions by half a day. Points show MSE reductions over the backbone for 100 settings. White diamonds show means, and bars show 95% bootstrap intervals.

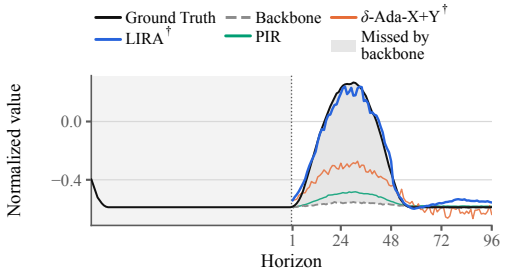



Figure 5: A forecast example on Weather. Comparison of LIRA with the backbone and other correction methods in one selected test window ($H = 96$). LIRA† recovers a peak missed by the other methods. Shading marks the look-back window, and † denotes online correction.

**RQ2: What should the head learn?**

**Learning the remaining error.** With the lookup fixed, we compare a head trained on raw backbone residuals (Lookup + raw head) with one trained on the error left after lookup correction (offline LIRA). Figure 6a includes lookup-only, head-only, and joint calendar + context baselines. Across 100 offline settings, learning the remaining error yields a larger mean MSE reduction over the backbone than raw-residual training or the joint model. The lookup and head work better together when they focus on different parts of the error. Appendix D.2 gives the comparison protocol.

**What the head corrects.** We split the fitted head correction $\Delta_t^{\text{head}}$ into two parts (Figure 6b). The mean across the $H$ leads is an offset that shifts the whole forecast up or down. The remaining zero-mean part changes the forecast shape. This decomposition splits the existing output without retraining. The offset and shape corrections are orthogonal, so the corresponding MSE gains over the lookup add up to the head's total gain. Averaged across settings, the offset accounts for about 70% of the MSE improvement from adding the head to the lookup. The head therefore contributes mainly by adjusting the forecast level in each window.

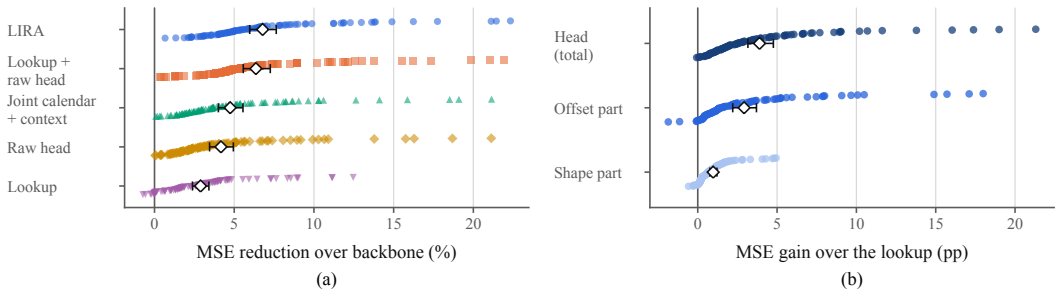



Figure 6: Head training and corrections. (a) Comparison of training the head on the original forecast error or the error left after lookup correction, with lookup-only, head-only, and joint calendar + context baselines. (b) The head's added MSE reduction, split into changes in forecast level and shape (percentage points relative to backbone MSE). Each point represents one of 100 offline settings using full correction history. White diamonds show means; bars show 95% bootstrap intervals.

**RQ3: How much history is needed?** We vary the most recent 10%–100% of correction history and refit every corrector, keeping the backbone and validation/test splits fixed and evaluating all methods offline (80 settings, Appendix D.1). Figure 7a shows that LIRA has the largest mean MSE reduction at each fraction. Using only the most recent 10% of history, LIRA exceeds the mean MSE reduction of the joint model, lookup alone, and raw-residual head alone, all fitted on the full history. Mean gains generally rise with more history, with a small dip from 50% to 75%.

**RQ4: What do online updates add?** We compare online and offline versions of the same fitted LIRA corrector, keeping the head and calibration coefficients fixed (100 settings, 25 per horizon, Appendix D.1). To prevent future-label leakage, residuals enter the online updates only after the corresponding full target window has been observed. Figure 7b shows that online LIRA improves average MSE for $H = 96, 192, 336$ after updates begin. For $H = 720$, feedback arrives later and fewer updates occur within the same evaluation period. The average gain at $H = 720$ remains near zero over the plotted interval.

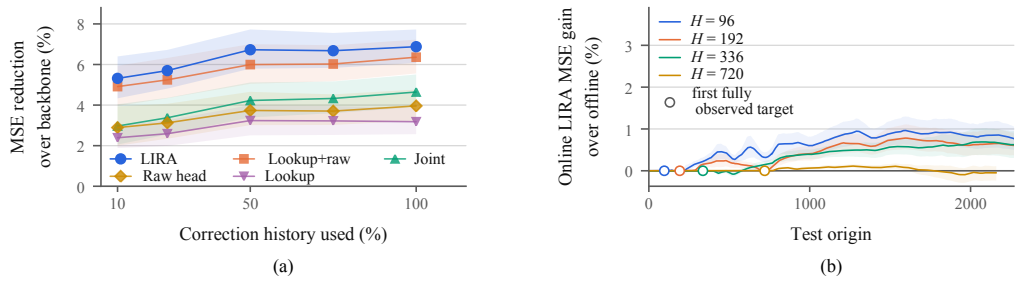



Figure 7: Correction history and online updates. (a) Comparison of LIRA with other correction methods using different amounts of past data. (b) Average improvement from online updates up to each test step, shown separately for each forecast horizon. Bands show 95% bootstrap intervals. Open circles mark when full targets first become available.

## 5 CONCLUSION

LIRA combines recurring error lookup with an input-conditioned remainder head, improving horizon-averaged accuracy across the evaluated task-specific and foundation backbones at low correction cost (Figure 3). The task-specific studies support earlier-to-later lookup transfer and remainder-target training, with offsets accounting for most of the fitted head's average additional gain. Mean improvement generally increases with correction history, and full-target feedback adds average gains through gradient-free statistical updates, most clearly at shorter horizons. These gains depend on recurring residual structure, sufficient correction history, and timely feedback; they do not establish robustness to arbitrary distribution shifts. Appendix E discusses the scope and limitations.

### AI USE STATEMENT

We used generative AI tools to assist with polishing the manuscript, including improving clarity, wording, and consistency of tone, and with literature retrieval and discovery, including identifying relevant related work. We also used these tools to assist with developing experimental code. The authors reviewed the AI-assisted text revisions and code and take full responsibility for the final manuscript and reported results, including all claims and references.

### ETHICS STATEMENT

This work studies post-hoc correction of time series forecasts using existing benchmarks covering transformer measurements, weather, electricity consumption, and solar-power generation. The study does not involve recruiting human participants or collecting new personal data. Forecasting errors can affect downstream decisions, and the reported benchmark improvements do not establish reliability in operational settings. Applications require evaluation under their intended conditions.

### REPRODUCIBILITY STATEMENT

Section 3 defines the proposed method, and Algorithm 1 summarizes its fitting and prediction procedures. Appendix A specifies calibration, head training, hyperparameter selection, random seeds, and the timing of online feedback. Appendix C documents the datasets, chronological splits, preprocessing, backbone configurations, baseline protocols, and metric aggregation. Appendix D provides the settings for the research question analyses.

## REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023a.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023b.

Harry R Glahn and Dale A Lowry. The use of model output statistics (mos) in objective weather forecasting. *Journal of Applied Meteorology and Climatology*, 11(8):1203–1211, 1972.

Shivam Grover and Ali Etemad. Shift-aware test time adaptation and benchmarking for time-series forecasting. In *Second Workshop on Test-Time Adaptation: Putting Updates to the Test! at ICML 2025*, 2025.

Qihe Huang, Zhengyang Zhou, Kuo Yang, Zhongchao Yi, Xu Wang, and Yang Wang. Timebase: The power of minimalism in efficient long-term time series forecasting. In *ICML*, 2025.

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.

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. *arXiv preprint arXiv:2605.20119*, 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.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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. *arXiv preprint arXiv:2601.20280*, 2026.

Shengsheng Lin, Weiwei Lin, Xinyi Hu, Wentai Wu, Ruichao Mo, and Haocheng Zhong. Cyclenet: Enhancing time series forecasting through modeling periodic patterns. *Advances in Neural Information Processing Systems*, 37:106315–106345, 2024.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International conference on learning representations*, volume 2024, pp. 11116–11140, 2024.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. *arXiv preprint arXiv:2502.00816*, 2025.

Zhiding Liu, Mingyue Cheng, Guanhao Zhao, Jiqian Yang, Qi Liu, and Enhong Chen. Improving time series forecasting via instance-aware post-hoc revision. *Advances in Neural Information Processing Systems*, 38:35604–35629, 2026.

Heitor R Medeiros, Hossein Sharifi-Noghabi, Gabriel L Oliveira, and Saghar Irandoust. Accurate parameter-efficient test-time adaptation for time series forecasting. *arXiv preprint arXiv:2506.23424*, 2025.

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://arxiv.org/abs/2211.14730.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. TimeMixer: Decomposable multiscale mixing for time series forecasting. In *International Conference on Learning Representations*, 2024a. URL https://arxiv.org/abs/2405.14616.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. *Advances in neural information processing systems*, 37:469–498, 2024b.

Jianxiang Xie, Yuncheng Hua, Mingyue Cheng, Flora Salim, and Hao Xue. Causality-inspired safe residual correction for multivariate time series. *arXiv preprint arXiv:2512.22428*, 2025.

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. URL https://www.sciencedirect.com/science/article/pii/S0925231201007020.

# A Implementation Details

We fit one corrector per dataset–backbone–horizon setting: estimate residual statistics, calibrate the lookup, train the remainder head, and calibrate the head. Test-time updates change only residual statistics. Algorithm 1 summarizes the procedure; Table 5 lists numerical settings.

## A.1 Inputs and Error Lookup

Inputs are the observed history, original backbone forecast, and forecast timestamp. Residuals are target minus original forecast. All values use training-fitted per-channel standardization; the backbone stays frozen.

The daily cycle has $S = 24$, 96, or 144 steps for hourly, 15-minute, or 10-minute data. Training residuals define the cycle, lead, and interaction patterns in Eq. (4). The stored joint table uses issue-time position and lead; adding the lead modulo $S$ recovers target-time position. Every cell used in fitting must contain observations.

We center the lead pattern within four consecutive, equal-sized horizon groups before applying its coefficient (Eq. (5)), consistently in target construction, validation, and prediction. The global residual mean constructs the patterns but is not a separate correction; the head learns forecast-level offsets. Appendix C.5 evaluates this centering.

## A.2 Calibration and Selection

**Calibrate the lookup first.** For each channel and horizon group, fit three coefficients in $[0, 1.5]$ by minimizing validation squared error. Online calibration replays validation from training statistics for each candidate update rate and selects the lowest-MSE rate and lookup. Offline calibration fixes the rate to zero. Freeze lookup coefficients before head training.

**Reduce unstable coefficients.** Refit the coefficients while leaving out one contiguous validation block at a time. Coefficients that change more across these fits are shrunk more strongly toward zero. The applied value is $\widehat{c}\widehat{c}^2/(\widehat{c}^2 + v)$, where $\widehat{c}$ is the full-validation estimate and $v$ is its block-jackknife variance; a zero denominator gives zero. Use $K = \min(20, \max(5, \lfloor N_{\text{val}}/(2H) \rfloor))$ blocks, where $N_{\text{val}}$ counts validation forecast origins. The variance is $(K-1)/K$ times the sum of squared deviations of the block-deletion estimates from their mean. Lookup features and head predictions stay fixed during these refits.

**Calibrate the head second.** For each trained head, fit one coefficient per channel and horizon group to the validation error remaining after lookup correction. The coefficient is bounded between 0 and 1 and uses the same shrinkage rule. Select head weight decay by the complete predictor's mean validation MSE over three seeds. Ties select the smaller update rate or weight decay. Offline and online variants are fitted and selected separately; an offline table entry is therefore not obtained by simply disabling updates in the online model.

**Numerical solver.** The lookup fit enumerates coefficients that are free or at either bound. If the regression Gram matrix has condition number above $10^6$, add a ridge term equal to $10^{-6}$ times its mean diagonal; otherwise use no ridge. Free-coordinate solves also add diagonal jitter equal to $10^{-12}$ times the trace of the ridge-adjusted matrix. Statistics and coefficient fits use float64.

## A.3 Training the Remainder Head

The shared linear head takes each channel's last $L$ observations and original $H$-step forecast, targeting the residual left after lookup correction.

**Construct held-out lookup targets.** Remove the final $H$ cached training origins, then divide the remaining origins into five contiguous folds. For each fold, estimate lookup statistics from the other training origins, excluding an additional $H$ origins on both sides of the fold. This prevents the fold's target observations from entering its own lookup statistics. Subtract this held-out lookup correction to obtain the head's training targets. The validation-fitted lookup coefficients are reused in all folds;

only the lookup statistics are cross-fitted. This offline construction can use both earlier and later training origins outside the excluded interval.

**Fit and stop training.** Train with Huber loss (threshold 1) and Adam (Kingma & Ba, 2014), with weight decay on both weights and biases. Early stopping uses the final 10% of the usable training origins, with $H$ origins removed on each side of the split boundary. If this tail is too short, reserve the final $H$ origins for stopping and leave a $2H$-origin gap before it. Keep the selected checkpoint without retraining on validation data. Each batch contains $\max(1, \mathrm{round}(4096/C))$ complete windows, including every channel.

Lookup targets are constructed before the inner stopping split. Some training targets can therefore use lookup statistics from the stopping tail outside the fold exclusion. The tail stops head optimization; it is not an independent holdout for the whole fitting procedure. Test targets never enter offline fitting.

Table 4: Which data are used at each fitting stage.

| Stage | Data | Quantities fitted or updated |
|---|---|---|
| Estimate lookup | Training residuals | Residual statistics |
| Calibrate lookup | Validation | Lookup coefficients and update rate |
| Build head targets | Training, with fold exclusions | Held-out lookup statistics |
| Train head | Training; inner tail for stopping | Shared linear weights and biases |
| Calibrate and select head | Validation | Head coefficients and weight decay |
| Online test prediction | Fully observed past targets | Residual statistics only |

### A.4 Online Feedback and Test Prediction

Process forecast origins in chronological order. A forecast issued at origin $s$ is eligible for feedback at origin $t$ only if $s+H \le t$: its entire target window must have been observed. Compute its residual using the saved original backbone forecast, without making another backbone call.

Collect eligible forecasts in issue order and update after each batch of $S$ forecasts. Move each global, lead, and observed cycle mean toward the corresponding batch mean using the selected update rate. An unobserved cycle entry is unchanged. For regularly sampled origins, each joint-table address is visited once per batch and receives the same moving-average update. If an address occurs more than once, the implementation sums its residual deviations from the old value, rather than averaging them or updating sequentially. All deviations in a batch use the pre-update state.

Apply all available batches before producing the next forecast. Read the current lookup correction, add the calibrated head correction, and save the original forecast for later feedback. The backbone, head, and calibration coefficients remain fixed throughout test prediction.

Evaluation carries validation statistics and pending feedback into test. Fresh deployment starts from training statistics with an empty queue. Offline statistics stay fixed.

### A.5 Numerical Settings

The head contains $H(L + H + 1)$ trained parameters. Calibration adds $4CQ$ coefficients, while lookup statistics require $C(1 + S + H + SH)$ stored values. The backbone and saved-forecast queue are separate from these counts. Appendix C.9 gives the boundaries used in the efficiency comparison.

**Remainder head objective.** Let $\mathcal{U}_{\mathrm{fit}}$ be the inner fitting origins and let the parameter vector $\theta$ stack the entries of $W$ and $\beta$ from Eq. (7). With $r_t^{\mathrm{rem}} = r_t - \Delta_t^{\mathrm{lookup,oof}}$, weight decay $\lambda \ge 0$, and Huber threshold $\kappa = 1$, we optimize

$$\min_{\theta} \frac{1}{|\mathcal{U}_{\mathrm{fit}}|HC} \sum_{t \in \mathcal{U}_{\mathrm{fit}}} \sum_{h=1}^{H} \sum_{j=1}^{C} \ell_{\mathrm{Huber}}\left(r_{t,h,j}^{\mathrm{rem}} - [W z_{t,j} + \beta]_h\right) + \frac{\lambda}{2}\|\theta\|_2^2. \tag{10}$$

The loss is $\ell_{\mathrm{Huber}}(e) = e^2/2$ for $|e| \le \kappa$ and $\kappa(|e| - \kappa/2)$ otherwise.

Table 5: Settings shared across LIRA runs. Selection uses validation only.

| Setting | Value |
|---|---|
| Lookback / forecast horizons | $L = 96$ / $H \in \{96, 192, 336, 720\}$ |
| Horizon groups / training folds | 4 / 5 |
| Daily cycle | 24, 96, or 144 steps, depending on sampling interval |
| Online update-rate candidates | $0.001, 0.01, 0.05$; offline uses 0 |
| Lookup / head coefficient bounds | $[0, 1.5]$ / $[0, 1]$ |
| Head optimizer / loss | Adam, learning rate $10^{-3}$ / Huber, threshold 1 |
| Maximum epochs / stopping patience | 50 / 5; minimum improvement $10^{-7}$ |
| Weight-decay candidates | $0, 0.001, 0.01, 0.1, 0.3$ |
| Batch size | $\max(1, \mathrm{round}(4096/C))$ windows |
| Training seeds | 0, 1, 2; average metrics, not predictions |
| Numerical precision | Statistics and calibration: float64; head and lookup outputs: float32 |

## B ALGORITHMIC SUMMARY

Algorithm 1 summarizes fitting and prediction; Appendix A gives settings and fold exclusions.

**Algorithm 1:** LIRA fitting and prediction. Online feedback uses only complete target windows.

**Input:** Frozen backbone; training and validation data; horizon $H$; daily cycle length $S$.
**Output:** Corrected forecasts.

**Fit the corrector**
Save the backbone's forecasts and residuals on training and validation data
Estimate the lookup statistics from usable training residuals
Use validation to select lookup coefficients and the update rate (zero for offline correction)
Build head targets: training residual minus the lookup correction estimated with that fold and its overlap margins excluded
Train candidate shared linear heads; stop using the inner training tail
Calibrate each head on validation and select weight decay by mean validation MSE across seeds
Keep the selected head, calibration coefficients, and update rate fixed

**Predict on the test stream**
Replay validation with the selected rate; carry lookup statistics and pending feedback into test
**foreach** *forecast origin in time order* **do**
  **if** *online correction is enabled* **then**
    Collect past forecasts whose entire target windows are now observed
    For every complete batch of $S$ eligible forecasts, compute residuals from the saved original forecasts and update only the lookup statistics
  **end**
  Obtain the new backbone forecast $\widehat{Y}_t$
  Read its lookup correction $\Delta_t^{\mathrm{lookup}}$ and compute the calibrated head correction $\Delta_t^{\mathrm{head}}$
  Return $\widehat{Y}_t + \Delta_t^{\mathrm{lookup}} + \Delta_t^{\mathrm{head}}$
  Save the original backbone forecast for future feedback
**end**

For fresh deployment, initialize statistics from training data with an empty feedback queue. Each seed yields a separate head; reported scores average their metrics.

## C DATASETS AND EVALUATION PROTOCOLS

Dataset counts below come from the CSV files used to create the forecast caches.

### C.1 DATASETS, SPLITS, AND PREPROCESSING

We use seven multivariate datasets: ETTh1, ETTh2, ETTm1, ETTm2, Weather, ECL, and Solar. The ETT datasets contain transformer temperature and load measurements; Weather contains mete-

orological variables; ECL contains electricity-consumption series; Solar contains solar-power series. All numeric variables are forecast. The date column supplies time information and is not a forecast variable.

Table 6: Dataset sizes and chronological splits. Variables exclude the date column. Timestamp and split counts are rows before forecast windows are formed.

| Dataset | Variables | Timestamps | Interval | Train | Validation | Test |
|---|---|---|---|---|---|---|
| ETTh1 | 7 | 17,420 | 60 min | 8,640 | 2,880 | 2,880 |
| ETTh2 | 7 | 17,420 | 60 min | 8,640 | 2,880 | 2,880 |
| ETTm1 | 7 | 69,680 | 15 min | 34,560 | 11,520 | 11,520 |
| ETTm2 | 7 | 69,680 | 15 min | 34,560 | 11,520 | 11,520 |
| Weather | 21 | 52,696 | 10 min | 36,887 | 5,270 | 10,539 |
| ECL | 321 | 26,304 | 60 min | 18,412 | 2,632 | 5,260 |
| Solar | 137 | 52,560 | 10 min | 36,792 | 5,256 | 10,512 |

**Chronological splits.** ETT uses 12/4/4 blocks of 30 days for training/validation/test: the first 14,400 hourly or 57,600 quarter-hourly rows; later rows are unused. Other datasets use 70%/10%/20% splits, rounding training and test counts down and assigning the remainder to validation (Table 6).

**Forecast windows and scaling.** Windows use $L = 96$, $H \in \{96, 192, 336, 720\}$, and stride one. History may cross split boundaries, but complete targets stay within their assigned split. Per-variable means and standard deviations are fitted on training rows and reused for all splits and correctors; metrics use this standardized scale. Appendix A.3 specifies additional training exclusions.

**Timestamps and the daily cycle.** The daily cycle has 24, 96, or 144 positions for 60-, 15-, or 10-minute sampling. The Weather file contains one repeated timestamp and one 100-minute gap. We retain those rows and use the configured 10-minute interval to advance the cycle from the issue timestamp. For windows crossing the irregularities, this computed target position can differ from the recorded target timestamp. Timestamp counts in Table 6 count rows, including the repeated Weather timestamp.

### C.2 Backbones and Comparison Coverage

**Task-specific models.** PatchTST, iTransformer, DLinear, TimeMixer, and TimeBase use dataset-specific checkpoints with a 96-step lookback. Every correction method starts from the same recorded checkpoint within a dataset–backbone–horizon setting. TimeBase uses upstream hyperparameters developed for a longer lookback; these results compare corrections to that checkpoint, rather than rank forecasting architectures. Table 1 covers all seven datasets. The online-adaptation comparison in Table 8 covers the four backbones other than TimeBase on the same datasets.

**Foundation models.** Table 2 uses Chronos-2 (`amazon/chronos-2`), Sundial (`thuml/sundial-base-128m`), TimesFM-2.5 (`google/timesfm-2.5-200m-pytorch`), and Toto-2.0 (`Datadog/Toto-2.0-22m`) on the four ETT datasets, Weather, and ECL. All four backbones stay frozen. Chronos-2 normalizes each channel within its context, requests quantiles 0.1/0.5/0.9, and uses the median forecast transformed back to the evaluation scale. Sundial averages 20 sampled trajectories using a fixed cache-generation seed. TimesFM-2.5 forecasts each channel as a univariate series with input normalization and its continuous quantile head, and uses its point (median) forecast. Toto-2.0 processes all channels of a window together and uses its median (0.5-quantile) forecast; a 768-step decode block covers each horizon in one pass. Every output corrector reuses the same cached realization. Corrector fitting uses task-specific historical labels, separately from backbone pretraining.

**Baseline configurations.** For $\delta$-Adapter, we follow the dataset-specific $\delta$ settings in the paper and the training and online hyperparameter settings in the released code (Liang et al., 2026). We fix $\delta = 0.01$ on ETT and $0.1$ on Weather, ECL, and Solar, and keep the backbone frozen. Validation is used for adapter early stopping.

PIR uses the released configurations. Because it has no released TimeBase configuration, its TimeBase runs reuse the PatchTST configuration for the corresponding dataset and horizon.

### C.3 Backbone Access and Feedback

Table 7 distinguishes corrections that can reuse saved forecasts from those that need additional backbone access. Offline LIRA fits a correction module around a frozen backbone. The input/output Ada-X+Y comparison is online; PIR jointly trains the backbone and its revision modules.

Table 7: Comparison interfaces. "Extra calls" refers to backbone calls beyond the saved original forecasts; "gradient" means differentiation through the backbone.

| Correction | Extra calls | Gradient | Backbone tuned | Feedback |
|---|---|---|---|---|
| $\delta$-Ada-Y | No | No | No | Offline |
| LIRA | No | No | No | Offline |
| LIRA$^{\dagger}$ | No | No | No | Online |
| $\delta$-Ada-X+Y$^{\dagger}$ | Yes | Yes | No | Online |
| PIR | Yes | Yes | Yes | Offline |
| $\delta$-Ada-Y$^{\dagger}$ | Yes | No | No | Online |

Online Ada-Y re-evaluates the frozen backbone for replay windows, but gradients are needed only for its output adapter. Ada-X+Y differentiates through the backbone to update its input adapter while keeping backbone parameters fixed.

For online LIRA, feedback is delayed until the complete target window is observed, as described in Appendix A.4. Both task-specific and foundation-model runs select the update rate from $\{0.001, 0.01, 0.05\}$ on validation. Offline runs use zero and are fitted separately.

Online Ada-Y and Ada-X+Y use batch size one and their respective optimizer settings from the released code. Updates use fully observed replay windows; offline Ada-Y keeps its fitted adapter fixed during testing.

### C.4 Comparison with Online Test-Time Adaptation

Table 8 compares LIRA with TAFAS (Kim et al., 2025), PETSA (Medeiros et al., 2025), DynaTTA (Grover & Etemad, 2025), and COSA-P (Im & Kwon, 2026) on four task-specific backbones and seven datasets. Results average four horizons and three seeds.

TAFAS, PETSA, and DynaTTA use their released code with a frozen backbone. We select their learning rate on validation from $\{0, 10^{-5}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$; zero disables updates. DynaTTA's adaptation-rate band is also selected on validation. Their evaluation uses period-based batches, updates from observed portions of the current target window and fully observed past windows, then re-forecasts.

COSA-P uses its official hyperparameters and periodicity-aware batch size on the same cached forecasts as LIRA. We disable partial-target adaptation and prediction adjustment, and update only after every target in a batch is observed. Thus, the methods use different feedback schedules.

### C.5 Why the Lead Pattern Is Centered

Replacing $\widetilde{b}^{\mathrm{lead}}$ with the uncentered $b^{\mathrm{lead}}$ and refitting the corrector causes a marked performance drop at $H = 720$ on ETTh2: test MSE increases by 3.8–6.1 percentage points of backbone MSE for eight of the nine backbones (Table 9). Most of this increase comes from errors in the forecast level. These results suggest that centering the lead pattern within each horizon group helps stabilize online correction by preventing this component from shifting the entire group's forecast level.

Table 8: **Comparison with online test-time adaptation.** MSE/MAE averaged over four forecast horizons for all 28 backbone–dataset pairs. Best correction scores are bold; second-best scores are underlined.

| | | | Online test-time adaptation | | | | Ours | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Backbone | Backbone | TAFAS | PETSA | DynaTTA | COSA-P | LIRA | LIRA† |
| ETTh1 | PatchTST | 0.451/0.450 | 0.454/0.454 | 0.449/0.452 | 0.457/0.456 | 0.449/0.450 | 0.440/0.445 | **0.439/0.444** |
| | iTransformer | 0.464/0.455 | 0.463/0.460 | 0.465/0.466 | 0.460/0.456 | 0.460/0.457 | 0.443/0.444 | **0.441/0.443** |
| | DLinear | 0.460/0.457 | 0.456/0.451 | 0.451/0.453 | 0.455/0.452 | 0.452/0.448 | 0.437/0.440 | **0.436/0.439** |
| | TimeMixer | 0.467/0.448 | 0.458/0.448 | 0.458/0.451 | 0.462/0.447 | 0.461/0.448 | 0.446/0.437 | **0.445/0.436** |
| ETTh2 | PatchTST | 0.385/0.413 | 0.378/0.407 | 0.382/0.410 | 0.380/0.408 | 0.384/0.411 | **0.365/0.398** | **0.365/0.398** |
| | iTransformer | 0.383/0.407 | 0.383/0.407 | 0.383/0.407 | 0.383/0.407 | 0.385/0.408 | **0.364**/0.394 | 0.365/**0.393** |
| | DLinear | 0.564/0.519 | 0.529/0.494 | 0.496/0.480 | 0.551/0.509 | 0.475/0.472 | **0.466/0.464** | 0.467/0.465 |
| | TimeMixer | 0.395/0.411 | 0.395/0.411 | 0.395/0.411 | 0.394/0.411 | 0.391/0.411 | **0.373/0.399** | 0.374/**0.399** |
| ETTm1 | PatchTST | 0.389/0.403 | 0.386/0.405 | 0.388/0.404 | 0.387/0.405 | 0.384/0.402 | 0.369/0.397 | **0.359/0.383** |
| | iTransformer | 0.407/0.411 | 0.391/0.410 | 0.393/0.411 | 0.395/0.408 | 0.394/0.409 | 0.385/0.401 | **0.373/0.388** |
| | DLinear | 0.404/0.408 | 0.392/0.400 | 0.391/0.400 | 0.396/0.402 | 0.393/0.402 | 0.365/0.387 | **0.360/0.382** |
| | TimeMixer | 0.383/0.398 | 0.376/0.396 | 0.380/0.401 | 0.379/0.396 | 0.380/0.398 | 0.370/0.392 | **0.360/0.381** |
| ETTm2 | PatchTST | 0.290/0.334 | 0.289/0.334 | 0.290/0.334 | 0.290/0.334 | 0.299/0.340 | 0.275/0.323 | **0.273/0.322** |
| | iTransformer | 0.291/0.334 | 0.291/0.334 | 0.291/0.334 | 0.291/0.335 | 0.301/0.341 | 0.272/0.320 | **0.271/0.318** |
| | DLinear | 0.355/0.402 | 0.297/0.349 | 0.295/0.348 | 0.327/0.375 | 0.325/0.365 | **0.287/0.344** | **0.287/0.344** |
| | TimeMixer | 0.277/0.324 | 0.275/0.323 | 0.276/0.323 | 0.275/0.323 | 0.283/0.328 | 0.261/0.314 | **0.260/0.312** |
| Weather | PatchTST | 0.258/0.280 | 0.249/0.287 | 0.275/0.291 | 0.259/0.293 | 0.251/0.278 | 0.236/0.279 | **0.231/0.268** |
| | iTransformer | 0.260/0.281 | 0.250/0.282 | 0.260/0.286 | 0.257/0.291 | 0.253/0.279 | 0.239/0.277 | **0.235/0.270** |
| | DLinear | 0.265/0.317 | 0.262/0.313 | 0.268/0.309 | 0.265/0.317 | 0.257/0.302 | 0.260/0.306 | **0.250/0.294** |
| | TimeMixer | 0.244/0.273 | 0.253/0.281 | 0.312/0.294 | 0.251/0.286 | 0.238/0.272 | 0.228/0.272 | **0.222/0.263** |
| ECL | PatchTST | 0.204/0.294 | 0.198/0.288 | 0.199/0.289 | 0.199/0.289 | 0.204/0.294 | **0.192/0.278** | **0.192/0.278** |
| | iTransformer | 0.180/0.270 | 0.178/0.270 | 0.181/0.273 | 0.178/0.271 | 0.180/0.270 | **0.175/0.267** | **0.175/0.267** |
| | DLinear | 0.225/0.319 | 0.222/0.316 | 0.224/0.317 | 0.224/0.319 | 0.225/0.318 | 0.209/0.297 | **0.208/0.295** |
| | TimeMixer | 0.185/0.275 | 0.185/0.275 | 0.186/0.278 | 0.185/0.276 | 0.186/0.275 | **0.182/0.272** | **0.182/0.272** |
| Solar | PatchTST | 0.269/0.307 | 0.266/0.315 | 0.265/0.313 | 0.268/0.326 | 0.269/0.307 | 0.241/0.281 | **0.226/0.270** |
| | iTransformer | 0.241/0.271 | 0.240/0.273 | 0.241/0.280 | 0.238/0.275 | 0.241/0.271 | 0.228/0.273 | **0.218/0.260** |
| | DLinear | 0.330/0.401 | 0.330/0.401 | 0.330/0.401 | 0.330/0.401 | 0.330/0.401 | 0.287/0.347 | **0.276/0.333** |
| | TimeMixer | 0.218/0.289 | 0.216/0.287 | 0.209/0.278 | 0.212/0.284 | 0.217/0.288 | 0.216/0.287 | **0.201/0.272** |

Table 9: **Failure cases of the uncentered lead pattern.** The online settings at $H = 720$, among the 213 refitted, in which replacing $\widetilde{b}^{\text{lead}}$ (Eq. 5) by the uncentered $b^{\text{lead}}$ in the held-out head targets, validation, and prediction raises the test MSE by more than 0.5 pp of backbone MSE. Selected update rate and test MSE with each pattern; change of the test MSE and of its per-window level and shape parts, and change of the lookup-only level error on validation. Changes are uncentered minus centered, in percentage points of the backbone MSE.

| | | | Update rate | | Test MSE | | Test MSE change | | | Validation |
|---|---|---|---|---|---|---|---|---|---|---|
| Backbone | Data | $H$ | $\widetilde{b}^{\text{lead}}$ | $b^{\text{lead}}$ | $\widetilde{b}^{\text{lead}}$ (ours) | $b^{\text{lead}}$ | Total | Level | Shape | level change |
| PatchTST | ETTh2 | 720 | 0.001 | 0.05 | **0.407** | 0.433 | +6.06 | +4.89 | +1.18 | −6.99 |
| iTransformer | ETTh2 | 720 | 0.01 | 0.05 | **0.400** | 0.423 | +5.50 | +4.81 | +0.69 | −8.55 |
| TimeBase | ETTh2 | 720 | 0.05 | 0.05 | **0.425** | 0.447 | +4.98 | +5.20 | −0.21 | −7.40 |
| Chronos-2 | ETTh2 | 720 | 0.01 | 0.05 | **0.428** | 0.449 | +4.82 | +4.32 | +0.50 | −10.53 |
| Sundial | ETTh2 | 720 | 0.01 | 0.05 | **0.422** | 0.442 | +4.69 | +4.12 | +0.57 | −10.48 |
| TimesFM-2.5 | ETTh2 | 720 | 0.01 | 0.05 | **0.433** | 0.453 | +4.51 | +4.21 | +0.31 | −10.23 |
| CycleNet | ETTh2 | 720 | 0.001 | 0.05 | **0.416** | 0.435 | +4.14 | +3.28 | +0.86 | −7.33 |
| TimeMixer | ETTh2 | 720 | 0.01 | 0.05 | **0.414** | 0.432 | +3.75 | +3.20 | +0.56 | −8.01 |
| TimeBase | ETTm1 | 720 | 0.01 | 0.05 | **0.417** | 0.424 | +1.42 | +0.05 | +1.38 | −1.38 |
| TimesFM-2.5 | ECL | 720 | 0.001 | 0.01 | **0.244** | 0.247 | +0.90 | +0.19 | +0.72 | −0.46 |
| TimeMixer | Solar | 720 | 0.05 | 0.05 | **0.220** | 0.222 | +0.89 | −0.10 | +0.99 | −0.35 |
| Sundial | ECL | 720 | 0.01 | 0.05 | **0.231** | 0.232 | +0.70 | +0.11 | +0.59 | −0.97 |

### C.6 METRICS AND AGGREGATION

For each setting and seed, compute MSE and MAE over all forecast origins, leads, and variables in the test split. Average the three seed scores first, then give each of the four horizons equal weight. Forecasts from different seeds are not ensembled. Table cells show MSE/MAE rounded to three decimals; best and second-best values are marked separately for the two metrics.

The percentage summaries of the main performance tables use the displayed unit of comparison: first average MSE over horizons for each backbone–dataset pair, calculate its percentage reduction from the backbone, then average equally across pairs. The RQ analyses instead calculate each reduction within a dataset–backbone–horizon setting and average across settings. These two aggregation orders can give different percentages.

RQ confidence intervals resample settings, rather than treating overlapping forecast windows as independent samples. Training-seed variation, backbone-checkpoint variation, and uncertainty over time are different quantities; the reported intervals measure variation across the evaluated settings.

### C.7 Seed Variation

Table 10 reports the standard deviation of the test MSE over the three training seeds, averaged for each backbone. In every setting, each seed of offline and online LIRA has lower MSE than the backbone.

Table 10: **Seed variation.** Standard deviation of the test MSE over three training seeds ($\times 10^{-3}$), averaged for each backbone over the datasets and horizons of Table 1.

| | Baselines | | | | Ours | |
|---|---|---|---|---|---|---|
| Model | $\delta$-Ada-Y | $\delta$-Ada-Y$^\dagger$ | $\delta$-Ada-X+Y$^\dagger$ | PIR | LIRA | LIRA$^\dagger$ |
| PatchTST | 0.28 | 0.27 | 0.24 | 2.42 | 0.55 | 0.45 |
| iTransformer | 0.27 | 0.29 | 0.34 | 2.11 | 0.56 | 0.53 |
| DLinear | 0.40 | 0.24 | 0.37 | 3.04 | 2.99 | 2.98 |
| TimeMixer | 0.19 | 0.29 | 0.28 | 1.21 | 0.25 | 0.23 |
| TimeBase | 0.24 | 0.25 | 0.20 | 1.29 | 0.79 | 0.80 |

### C.8 Lookback Length

We retrain PatchTST and iTransformer with $L = 336$ on ETTh1, ETTm2, and Weather, keeping their other training settings, and refit LIRA with a 336-step head input. Table 11 compares the results with the $L = 96$ backbones of Table 1. Across the six pairs, the mean MSE reduction of LIRA$^\dagger$ falls from $6.78\%$ at $L = 96$ to $4.63\%$ at $L = 336$, and its range narrows from 2.6–$10.7\%$ to 3.6–$5.6\%$. The reduction shrinks most on Weather, where the longer lookback also improves the backbone most, and grows for PatchTST on ETTh1, whose $L = 336$ backbone has higher error than its $L = 96$ counterpart.

Offline and online LIRA lower the MSE in all 24 $L = 336$ settings.

Table 11: **Lookback length.** MSE/MAE averaged over four forecast horizons for backbones trained with lookback $L = 96$ (the setting of Table 1) and $L = 336$. Red. is the MSE reduction (%) of LIRA$^\dagger$ relative to the backbone with the same lookback. The lowest MSE and MAE within each lookback are bold.

| | | $L = 96$ | | | | $L = 336$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | Dataset | Backbone | LIRA | LIRA$^\dagger$ | Red. | Backbone | LIRA | LIRA$^\dagger$ | Red. |
| PatchTST | ETTh1 | 0.451/0.450 | 0.440/0.445 | **0.439/0.444** | 2.6 | 0.489/0.479 | 0.462/**0.462** | **0.461/0.462** | 5.6 |
| | ETTm2 | 0.290/0.334 | 0.275/0.323 | **0.273/0.322** | 5.8 | 0.270/0.330 | **0.258**/0.320 | **0.258/0.319** | 4.4 |
| | Weather | 0.258/0.280 | 0.236/0.279 | **0.231/0.268** | 10.7 | 0.234/0.272 | 0.224/0.269 | **0.223/0.268** | 4.8 |
| iTransformer | ETTh1 | 0.464/0.455 | 0.443/0.444 | **0.441/0.443** | 4.9 | 0.450/0.457 | **0.434/0.446** | **0.434/0.446** | 3.6 |
| | ETTm2 | 0.291/0.334 | 0.272/0.320 | **0.271/0.318** | 6.9 | 0.272/0.329 | **0.260**/0.318 | **0.260/0.317** | 4.5 |
| | Weather | 0.260/0.281 | 0.239/0.277 | **0.235/0.270** | 9.7 | 0.238/0.272 | **0.227**/0.270 | **0.227/0.268** | 4.9 |

### C.9 Correction Cost Accounting

This appendix details the analytical counts in Figure 3, including storage for lookup statistics and retrieved examples. Offline and online LIRA have the same prediction cost.

The counts fix $L = 96$, $H = 336$, retrieval length $N = 8192$, daily cycle $S = 24$, and four horizon groups, while varying the number of channels over $\{7, 21, 137, 321\}$. Ada-Y and Ada-X+Y use hidden width 512. PIR uses embedding and feed-forward widths 128, one layer, and top-50 retrieval, following its PatchTST–ETTh1–$H = 336$ configuration.

A MAC is one multiply–accumulate. Counts cover correction for one forecast and exclude backbone calls, fitting, online updates, normalization, activations, retrieval selection, lookup combination, and memory movement; they are not runtime measurements.

Table 12: **Analytical correction costs at $C = 321$.** Other dimensions follow Figure 3. Core scalars add fitted weights, calibration coefficients, lookup statistics, and retrieval values. Core MiB uses FP32 for neural weights and retrieval values and FP64 for LIRA's calibration and lookup statistics ($1\,\mathrm{MiB} = 2^{20}$ bytes). Backbone storage and runtime buffers are excluded.

| Method | MMACs | Weights + calib. | Lookup state | Retrieval state | Core scalars | Core MiB |
|---|---|---|---|---|---|---|
| LIRA | 46.594 | 150,624 | 2,704,425 | 0 | 2,855,049 | 21.2 |
| PIR | 379.439 | 382,298 | 0 | 1,136,001,024 | 1,136,383,322 | 4,335.0 |
| Ada-Y | 110.707 | 110,815,568 | 0 | 0 | 110,815,568 | 422.7 |
| Ada-X+Y | 142.524 | 142,665,136 | 0 | 0 | 142,665,136 | 544.2 |

Table 12 separates fitted weights, biases, and calibration coefficients from stored statistics and examples. Lookup storage is $C(1 + S + H + SH)$; PIR retrieval storage is $CN(L + H)$, comprising input–target pairs. Byte counts use the implementation dtypes: FP32 neural weights and PIR retrieval values, and FP64 LIRA lookup statistics and calibration coefficients. The listed values occupy 21.2 MiB for LIRA and 422.7 MiB for Ada-Y, a storage ratio of about 20. These counts exclude backbone storage, auxiliary tensors, duplicate coefficient copies, feedback queues, temporary buffers, activations, optimizer state, and allocator overhead. They are analytical core-storage counts, not measurements of resident or peak memory. Among these methods, only PIR also trains backbone parameters.

Table 13 varies $H$ at $C = 321$. The head's MACs grow as $CH(L + H)$, while adapter MACs grow linearly in $H$ at fixed width. Ada-Y therefore costs 5.4 times as much at $H = 96$ but 1.3 times at $H = 720$, with equal costs near $H = 930$.

Table 13: **Correction arithmetic across horizons at $C = 321$.** Leading MMACs per forecast; $L$, $N$, and module widths follow Table 12.

| Method | $H = 96$ | $H = 192$ | $H = 336$ | $H = 720$ |
|---|---|---|---|---|
| LIRA | 5.9 | 17.8 | 46.6 | 188.6 |
| PIR-F, PIR | 345.9 | 359.3 | 379.4 | 433.0 |
| Ada-Y | 31.8 | 63.4 | 110.7 | 236.9 |
| Ada-X+Y | 63.6 | 95.2 | 142.5 | 268.7 |

### C.10 CYCLE-AWARE AND CALENDAR-INFORMED BACKBONES

Table 14: **Correction after learning cycles in the backbone.** CycleNet/Linear with instance normalization (Lin et al., 2024) learns recurring patterns during training (backbone seed 2024), after which we freeze the checkpoint and apply LIRA. Entries are MSE/MAE averaged over $H \in \{96, 192, 336, 720\}$ and three correction seeds at $L = 96$. A dagger denotes online residual-statistic updates after full targets are observed. Red. is the online MSE reduction (%); the lowest scores per row are bold, including ties. Online LIRA lowers horizon-averaged MSE on all six datasets, so learning cycles in the backbone still leaves correctable error.

| Dataset | CycleNet | +LIRA | +LIRA$^{\dagger}$ | Red. |
|---|---|---|---|---|
| ETTh1 | 0.432/**0.427** | 0.430/0.431 | **0.429**/0.429 | 0.8 |
| ETTh2 | 0.384/0.405 | **0.366**/0.394 | **0.366/0.393** | 4.7 |
| ETTm1 | 0.386/0.395 | 0.377/0.392 | **0.363/0.380** | 6.0 |
| ETTm2 | 0.272/0.315 | 0.266/0.312 | **0.264/0.310** | 2.9 |
| Weather | 0.254/0.279 | 0.240/0.284 | **0.233/0.271** | 8.6 |
| ECL | 0.170/0.261 | **0.167/0.260** | **0.167/0.260** | 1.7 |

Table 15: **Correction after training with calendar inputs.** TiDE (Das et al., 2023a) and TimeXer (Wang et al., 2024b) are trained from scratch with calendar features (backbone seed 2021) and then frozen. TiDE receives lookback and forecast-window calendar features, while TimeXer receives lookback calendar features as exogenous tokens. Entries are MSE/MAE averaged over $H \in \{96, 192, 336, 720\}$ and three correction seeds at $L = 96$. A dagger denotes online residual-statistic updates after full targets are observed. Red. is the online MSE reduction (%); the lowest scores per row are bold, including ties. Online LIRA lowers horizon-averaged MSE on all six model–dataset pairs, showing that calendar inputs do not eliminate the benefit of residual correction.

| Model | Dataset | Backbone | +LIRA | +LIRA$^{\dagger}$ | Red. |
|---|---|---|---|---|---|
| TiDE | ETTh1 | 0.447/0.434 | **0.434**/0.432 | **0.434/0.431** | 2.9 |
| | ETTh2 | 0.378/0.401 | **0.367/0.394** | **0.367/0.394** | 2.9 |
| | Weather | 0.273/0.291 | 0.253/0.297 | **0.243/0.277** | 10.8 |
| TimeXer | ETTh1 | 0.444/0.445 | **0.432/0.438** | **0.432/0.438** | 2.7 |
| | ETTh2 | 0.376/0.401 | **0.358/0.390** | **0.358/0.390** | 4.9 |
| | Weather | 0.241/0.271 | 0.227/0.270 | **0.222/0.260** | 7.8 |

### C.11 BACKBONE FINE-TUNING

Fine-tune updates the task-specific backbone weights offline, and Fine-tune† continues these updates on fully observed test targets. Both variants update all backbone weights from the same checkpoint used by LIRA, using Adam, MSE loss, and the original training batch size. We select the learning rate and epoch count (0–5, with zero retaining the checkpoint) by seed-mean MSE on the last 20% of validation origins, with selection-training targets ending before that held-out block. Fine-tune then refits on the training and validation targets and freezes the weights during testing. Fine-tune† continues from these weights, taking one pass over newly fully observed windows every $S$ forecast origins. The online learning rate is selected by streaming the same validation block, with zero included to allow no updates.

Table 16 compares these variants with LIRA. Backbone fine-tuning lowers MSE by $1.53\%$ offline and $1.66\%$ online on average across the 35 horizon-averaged backbone–dataset pairs. DLinear has the largest mean reductions, at $5.93\%$ offline and $7.66\%$ online across the seven datasets, compared with less than $1\%$ for each of the other backbones under either variant.

Table 16: **Backbone fine-tuning.** Comparison of fine-tuning the backbone with correcting its frozen forecasts using LIRA. MSE/MAE averaged over four forecast horizons and three seeds. † denotes online updates. Best and second-best method scores are bold and underlined.

| Dataset | Model | Backbone | Fine-tune | Fine-tune† | LIRA | LIRA† |
|---|---|---|---|---|---|---|
| ETTh1 | PatchTST | 0.451/0.450 | 0.455/0.453 | 0.453/0.451 | 0.440/0.445 | **0.439/0.444** |
| | iTransformer | 0.464/0.455 | 0.461/0.454 | 0.458/0.452 | 0.443/0.444 | **0.441/0.443** |
| | DLinear | 0.460/0.457 | 0.467/0.463 | 0.469/0.463 | 0.437/0.440 | **0.436/0.439** |
| | TimeMixer | 0.467/0.448 | 0.461/0.448 | 0.458/0.446 | 0.446/0.437 | **0.445/0.436** |
| | TimeBase | 0.463/0.429 | 0.465/0.430 | 0.465/0.430 | 0.441/0.429 | **0.438/0.428** |
| ETTh2 | PatchTST | 0.385/0.413 | 0.385/0.411 | 0.380/0.407 | **0.365/0.398** | **0.365/0.398** |
| | iTransformer | 0.383/0.407 | 0.384/0.407 | 0.382/0.407 | **0.364**/0.394 | 0.365/**0.393** |
| | DLinear | 0.564/0.519 | 0.457/0.463 | **0.427/0.437** | 0.466/0.464 | 0.467/0.465 |
| | TimeMixer | 0.395/0.411 | 0.395/0.412 | 0.390/0.408 | **0.373/0.399** | 0.374/**0.399** |
| | TimeBase | 0.405/0.421 | 0.409/0.423 | 0.408/0.421 | **0.382/0.405** | 0.383/**0.405** |
| ETTm1 | PatchTST | 0.389/0.403 | 0.390/0.403 | 0.388/0.401 | 0.369/0.397 | **0.359/0.383** |
| | iTransformer | 0.407/0.411 | 0.409/0.413 | 0.407/0.411 | 0.385/0.401 | **0.373/0.388** |
| | DLinear | 0.404/0.408 | 0.400/0.405 | 0.400/0.405 | 0.365/0.387 | **0.360/0.382** |
| | TimeMixer | 0.383/0.398 | 0.380/0.397 | 0.378/0.393 | 0.370/0.392 | **0.360/0.381** |
| | TimeBase | 0.429/0.419 | 0.419/0.410 | 0.419/0.410 | 0.372/0.392 | **0.359/0.382** |
| ETTm2 | PatchTST | 0.290/0.334 | 0.289/0.332 | 0.296/0.334 | 0.275/0.323 | **0.273/0.322** |
| | iTransformer | 0.291/0.334 | 0.288/0.331 | 0.289/0.332 | 0.272/0.320 | **0.271/0.318** |
| | DLinear | 0.355/0.402 | 0.301/0.361 | **0.286/0.336** | 0.287/0.344 | 0.287/0.344 |
| | TimeMixer | 0.277/0.324 | 0.276/0.323 | 0.306/0.334 | 0.261/0.314 | **0.260/0.312** |
| | TimeBase | 0.290/0.332 | 0.290/0.332 | 0.289/0.329 | 0.269/0.316 | **0.268/0.314** |
| Weather | PatchTST | 0.258/0.280 | 0.257/0.280 | 0.261/0.283 | 0.236/0.279 | **0.231/0.268** |
| | iTransformer | 0.260/0.281 | 0.259/0.279 | 0.258/0.278 | 0.239/0.277 | **0.235/0.270** |
| | DLinear | 0.265/0.317 | 0.265/0.316 | 0.260/0.306 | 0.260/0.306 | **0.250/0.294** |
| | TimeMixer | 0.244/0.273 | 0.244/0.273 | 0.244/0.273 | 0.228/0.272 | **0.222/0.263** |
| | TimeBase | 0.252/0.279 | 0.253/0.281 | 0.253/0.280 | 0.234/0.282 | **0.227/0.270** |
| ECL | PatchTST | 0.204/0.294 | 0.202/0.294 | 0.200/0.292 | **0.192/0.278** | **0.192/0.278** |
| | iTransformer | 0.180/0.270 | 0.177/0.268 | 0.177/0.268 | **0.175/0.267** | **0.175/0.267** |
| | DLinear | 0.225/0.319 | 0.210/0.297 | 0.210/0.297 | 0.209/0.297 | **0.208/0.295** |
| | TimeMixer | 0.185/0.275 | 0.185/0.275 | 0.185/0.275 | **0.182/0.272** | **0.182/0.272** |
| | TimeBase | 0.227/0.295 | 0.226/0.295 | 0.227/0.295 | **0.211/0.290** | **0.211/0.290** |
| Solar | PatchTST | 0.269/0.307 | 0.259/0.302 | 0.259/0.299 | 0.241/0.281 | **0.226/0.270** |
| | iTransformer | 0.241/0.271 | 0.242/0.268 | 0.242/0.268 | 0.228/0.273 | **0.218/0.260** |
| | DLinear | 0.330/0.401 | 0.326/0.398 | 0.323/0.396 | 0.287/0.347 | **0.276/0.333** |
| | TimeMixer | 0.218/0.289 | 0.217/0.281 | 0.217/0.281 | 0.216/0.287 | **0.201/0.272** |
| | TimeBase | 0.385/0.375 | 0.385/0.375 | 0.384/0.376 | 0.270/0.326 | **0.266/0.320** |

## D SETTINGS FOR THE MAIN RQ EXPERIMENTS

### D.1 DATASETS, HORIZONS, AND AVAILABLE SETTINGS

The default set contains 100 combinations: ETTh1, ETTh2, ETTm1, ETTm2, and Weather; PatchTST, iTransformer, DLinear, TimeMixer, and TimeBase; and horizons 96, 192, 336, and 720.

The lookback is 96. Each experiment keeps the same backbone forecasts and chronological validation/test splits for the methods it compares.

- **RQ1: lookup transfer.** All 100 settings are evaluated offline on a later block of correction-training data (Figure 4).
- **RQ2: remainder learning and head decomposition.** Both analyses use all 100 settings with offline lookup statistics and the full usable correction history.
- **RQ3: correction history.** All methods use the same 80 settings supporting offline evaluation at history fractions 10%, 25%, 50%, 75%, and 100%. ETTh1 and ETTh2 at horizons 336 and 720 are excluded because the shortest histories leave some held-out lookup cells empty.
- **RQ4: online updates.** All 100 settings are used, with the same 25 averaged at each horizon throughout a common initial test interval. Online and offline runs share the fitted corrector and initial statistics; the head and calibration coefficients stay fixed.

### D.2 RQ2: Head Training Comparisons

Figure 6a uses all 100 settings with offline lookup statistics and the full usable correction history. Lookup + raw head and offline LIRA use the same fixed lookup. The former trains the head on raw backbone residuals, while the latter trains it on the error left after lookup correction. For both variants, we fit the head calibration coefficients $\gamma$ on validation data. The lookup alone and the raw-residual head alone are also included as baselines.

Joint calendar + context is a single linear head trained on raw backbone residuals. Its inputs are the recent observations, the original backbone forecast, and a one-hot encoding of the issue cycle position, with separate output weights for each forecast lead. This baseline models calendar and context jointly in one head.

## E Limitations

**User-specified cycle.** LIRA requires the user to specify a cycle length rather than discovering it automatically. Its effectiveness depends on whether this choice captures recurring residual patterns. Choosing a suitable cycle can be difficult when periodicity is unclear or changes over time.

**Cross-channel dependencies.** The head applies one shared linear map to each channel separately and does not explicitly model dependencies across channels. This may limit gains on datasets with strong cross-channel relationships. On ECL, for example, LIRA improves backbone accuracy, but the relative gains are smaller than on several other datasets. Better use of cross-channel information may help address this limitation.