FORECAST, GUIDE, REFINE: FUTURE STATE GUIDANCE FOR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Future target representations can complement observation space supervision in multivariate time series forecasting. However, we identify *Predicted State Deviation*: a future state predicted from encoded history can inaccurately represent the true future, making it unreliable for direct decoding into a final forecast. Yet such an imperfect prediction may still provide useful guidance. We propose FUSE, a future state guidance framework following a *forecast, guide, and refine* paradigm. FUSE first generates a base forecast in observation space and encodes it as a base state, while predicting a future state from encoded history. Rather than directly decoding the predicted state, FUSE learns how far to move the base state along or against the displacement toward it. The resulting refined state is decoded to observation space and combined with an adjusted reconstruction residual to preserve base forecast information. FUSE can also be directly applied to existing backbones without retraining or modifying the backbone. Experiments on eight benchmarks show competitive accuracy with a simple MLP and consistent average improvements across three backbones. Ablations validate state refinement and residual adjustment, while decomposition analyses reveal corrections in forecast mean, waveform scale, and residual shape.

1 INTRODUCTION

Multivariate time series forecasting predicts future observations from historical data and supports planning and decision making across diverse applications (Lim & Zohren, 2021). Recent forecasting research has explored attention-based architectures (Nie et al., 2023; Liu et al., 2024; Qiu et al., 2025) and convolutional networks (Wu et al., 2023; Luo & Wang, 2024). Linear and MLP-based models also achieve competitive forecasting accuracy (Zeng et al., 2023; Wang et al., 2024a; Huang et al., 2025). These methods follow a common paradigm of mapping historical observations to future predictions, as illustrated in Figure 1(a). Complementary to backbone design, another direction is to use future targets during training to learn how to improve these existing forecasts.

Future targets can provide training supervision through encoded representations, rather than errors measured in observation space. Such supervision has been explored in sequence generation through representations of future context (Serdyuk et al., 2018), and in video learning through prediction of future features (Han et al., 2020). In time series forecasting, recent work aligns predictive representations with representations learned by reconstructing future targets (Hu et al., 2026). Another direction predicts future states in a learned latent space and decodes them into observation-space forecasts (Yang et al., 2026). These approaches highlight the value of representations derived from future observations as prediction targets or alignment references during training. Building on this, we first consider predicting future states from history under such supervision and decoding them directly into forecasts, but state prediction errors can limit forecasting accuracy. We therefore examine whether these imperfect predicted states can still provide guidance for refinement. This leads to the central research question: *how to use future predicted state to refine an existing forecast?*

To investigate this question, we encode the base forecast and the true future targets into the same encoded state space, obtaining the base and target states, respectively. We then compare the distances of the predicted and base states to the target state. However, the predicted state can lie farther from the target than the base state. We term this phenomenon *Predicted State Deviation*, as illustrated in Figure 1(b). Figure 2 shows this phenomenon across seven datasets with independently trained

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.

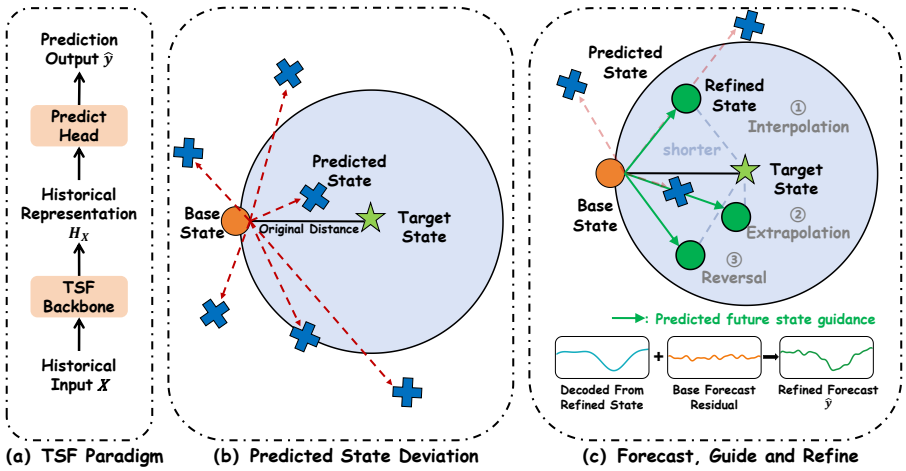

Figure 1: **Motivation and conceptual overview of FUSE.** (a) Conventional forecasting paradigm. (b) Illustration of *Predicted State Deviation*. (c) The *Forecast, Guide, and Refine* approach.

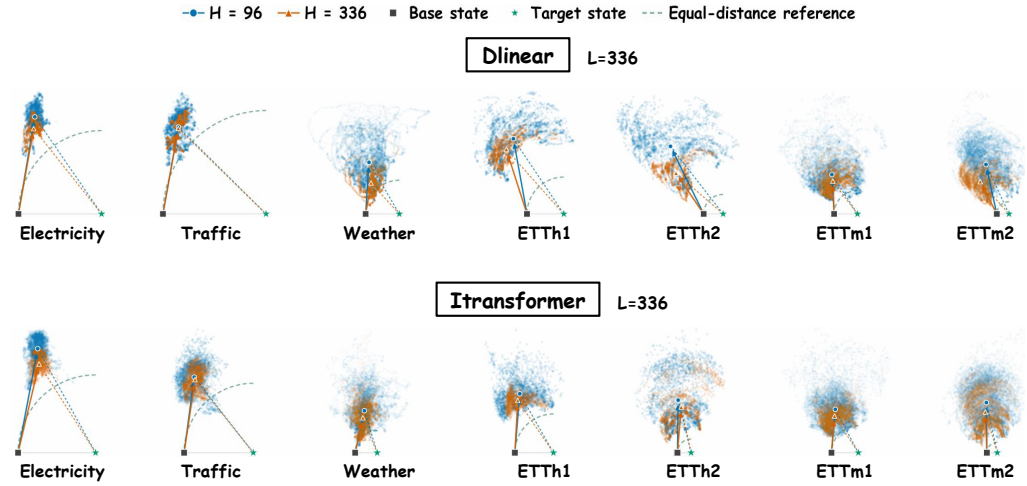

Figure 2: **Predicted State Deviation across datasets.** For each dataset, a pretrained, frozen MLP autoencoder defines the shared state space. Point represents test windows, with base–target distance normalized to one. The geometry and diagnostic protocol are detailed in Appendices A.1 and A.2.

DLinear (Zeng et al., 2023) and iTransformer (Liu et al., 2024) future state predictors. Most predicted states fall outside the equal-distance reference boundary. Yet an inaccurate predicted state can still provide useful guidance when its displacement from the base state is scaled or reversed. This motivates our central idea: *forecast, guide, and refine*. We use the displacement as guidance and learn how far to move the base state along or against it.

We propose **FUSE**, a **FU**ture **S**tate guidanc**E** framework for time series forecasting that implements this idea. FUSE first generates a base forecast from history and uses its encoded state as the starting base state for refinement, building on the information already captured by the base forecaster. A future state predictor then estimates a future predicted state from encoded history in the same space. Its displacement from the base state provides guidance for refinement. A shared learned coefficient controls how far to move along or against this displacement, allowing interpolation between the states, extrapolation beyond the predicted state, and reversal in the opposite direction (Figure 1(c)). Since encoding and decoding may not fully recover the base forecast, FUSE also preserves and adjusts its reconstruction residual. Combining this residual with the decoded refined state produces

the final forecast. Additionally, FUSE can be directly applied to different pretrained base forecasters without modifying or jointly retraining the backbone. Our contributions are summarized as follows:

- We identify *Predicted State Deviation*, showing that a future state predicted from history under supervision from encoded future targets can be less accurate than the state encoded from an existing forecast.
- We propose FUSE, which converts predicted future states into guidance and learns state refinement and reconstruction residual adjustment for more effective forecast performance.
- Experiments on eight datasets demonstrate competitive forecasting accuracy and improvements across three backbones. Ablations clarify component roles, while decomposition analyses reveal corrections in mean, scale, and residual shape.

2 Related Work

Time Series Forecasting and Refinement. Forecasting architectures capture temporal patterns through patching and multiscale attention (Nie et al., 2023; Chen et al., 2024), and model multivariate dependencies through attention across variables (Liu et al., 2024; Wang et al., 2024b; Liu et al., 2025a). Linear and MLP-based alternatives emphasize decomposition and temporal mixing (Zeng et al., 2023; Wang et al., 2024a). Complementary to backbone design, historical retrieval provides reference trajectories (Han et al., 2025), while contextual or retrieved sequences can guide the revision of existing forecasts (Liu et al., 2025b). With frozen backbones, forecast corrections can be learned through bounded input-output adjustments (Liang et al., 2026) or adapted online using recently observed targets (Im & Kwon, 2026). FUSE instead uses predicted future states to guide refinement of an existing forecast, together with reconstruction residual adjustment, without requiring changes to the forecasting backbone.

Future Representations for Forecasting. Predictive representation learning uses representations of future observations (van den Oord et al., 2018; Han et al., 2020) or masked content in vision and language (Baevski et al., 2022; Assran et al., 2023) as supervisory targets. In time series forecasting, target reconstruction provides representations for aligning predictive features (Hu et al., 2026). Another line of work models temporal dynamics in an encoded space before decoding forecasts (Liu et al., 2023), with some methods directly supervising state prediction using encoded future targets (Yang et al., 2026). FUSE similarly learns future state predictions, but uses them to define a displacement relative to the encoded base forecast. It then learns how far to move along or against this displacement, using imperfect predicted states as guidance for forecast refinement rather than as direct targets.

3 Method

3.1 Problem Definition and Overview

Let $\boldsymbol{x}_t \in \mathbb{R}^C$ denote the observations of C variables at time step t. Given a historical window $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L]^\top \in \mathbb{R}^{L \times C}$, multivariate time series forecasting aims to predict $Y = [\boldsymbol{x}_{L+1}, \ldots, \boldsymbol{x}_{L+H}]^\top \in \mathbb{R}^{H \times C}$, where L and H denote the lookback length and forecast horizon. A base forecaster F first produces an initial prediction $Y_0 = F(X)$.

FUSE then follows three steps: *forecast, guide, and refine*, as illustrated in Figure 3. A fixed encoder E maps the base forecast and true future into a shared state space, yielding the base state $S_0 = E(Y_0)$ and target state $S^\star = E(Y)$. Meanwhile, a future state predictor estimates $\widehat{S}$ from encoded history $E(X)$, supervised by $S^\star$ during training. Because $\widehat{S}$ may be an inaccurate refinement endpoint, FUSE instead uses the displacement $\widehat{S} - S_0$ as guidance. It learns how far to move the base state along or against this displacement, producing a refined state S_{ref}. Finally, S_{ref} is decoded and combined with an adjusted reconstruction residual of Y_0 to obtain the refined forecast Y_{ref}.

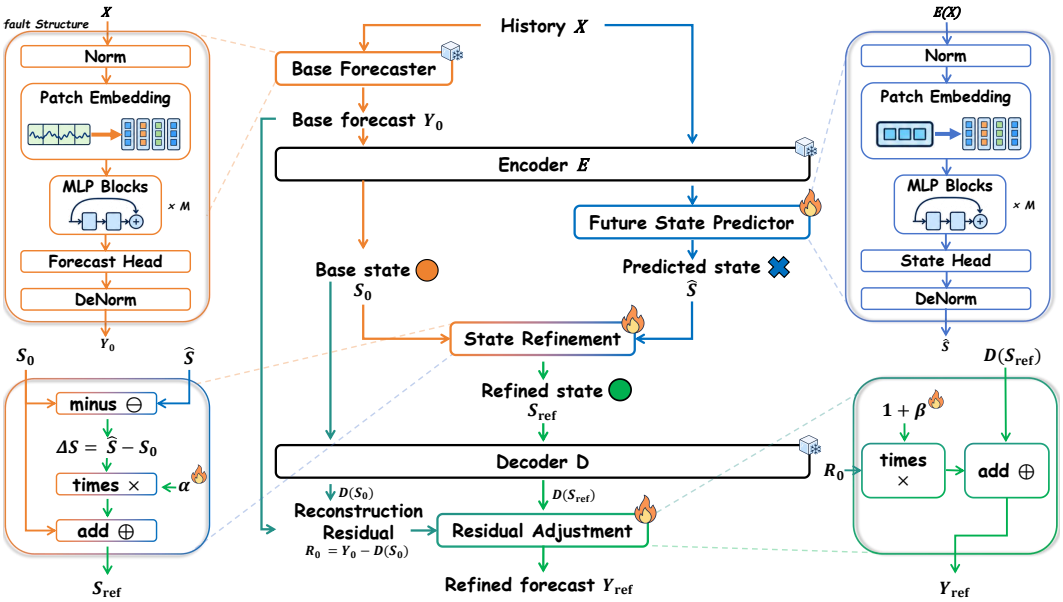

Figure 3: **Architecture of FUSE**. *Forecast*: produce the base forecast. *Guide*: predict a future state from history. *Refine*: adjust the base state and reconstruction residual using learned coefficients α and β. The same encoder and decoder are reused, with their weights frozen after pretraining.

3.2 Base Forecasting

We use a simple MLP as the default implementation, while allowing other forecasting backbones to serve as F. After normalizing X per channel using RevIN (Kim et al., 2022), we divide each channel into N_p temporal patches. Linear projection and positional encoding produce patch embeddings $U^{(0)} \in \mathbb{R}^{C \times N_p \times d_f}$, where d_f is the embedding dimension. These embeddings pass through M residual MLP blocks:

$$U^{(\ell)} = \mathrm{LN}_\ell\Big(U^{(\ell-1)} + \mathrm{MLP}_\ell\Big(U^{(\ell-1)}\Big)\Big), \qquad \ell = 1, \ldots, M, \tag{1}$$

where each MLP_ℓ consists of two linear layers with GELU activation and dropout between them, and LN_ℓ denotes layer normalization. A linear prediction head maps the flattened patch features of each channel to H future values. The predictions are then denormalized using the historical statistics to obtain Y_0.

3.3 Future State Guidance

To guide refinement of Y_0, we first represent the history and base forecast in a shared state space. We obtain this space using an autoencoder pretrained to reconstruct the multivariate observation at each time step (Yang et al., 2026). Its encoder E maps observations to states of dimension d, while its decoder D reconstructs observations from these states. Both remain frozen during FUSE training. Applying the same encoder to X and Y_0 gives

$$S_X = E(X) \in \mathbb{R}^{L \times d}, \qquad S_0 = E(Y_0) \in \mathbb{R}^{H \times d}. \tag{2}$$

From S_X, a compact future state predictor P_θ estimates $\widehat{S} = P_\theta(S_X) \in \mathbb{R}^{H \times d}$. It follows the normalization, patch embedding, and residual MLP structure in Section 3.2, with independent parameters. Its state head maps the flattened patch features to H future states, followed by denormalization using the statistics of S_X.

Since $\widehat{S}$ and S_0 correspond to the same future horizon in shared coordinates, we define their displacement as

$$\Delta S = \widehat{S} - S_0. \tag{3}$$

Rather than treating $\widehat{S}$ as a fixed endpoint, FUSE uses ΔS as guidance for refining the base state.

3.4 FORECAST REFINEMENT

Predicted State Deviation indicates that $\widehat{S}$ is not necessarily a reliable refinement endpoint. We therefore use a learnable coefficient α to scale ΔS before applying it to the base state:

$$S_{\mathrm{ref}} = S_0 + \alpha \Delta S. \tag{4}$$

Here, $\alpha \in \mathbb{R}$ is a scalar shared across inputs to each trained FUSE model. Its value determines interpolation ($0 < \alpha < 1$), extrapolation ($\alpha > 1$), or reversal ($\alpha < 0$). Although α is shared, ΔS varies across inputs, yielding input-dependent state corrections. The frozen decoder then maps S_{ref} back to the observation space.

Because reconstruction is imperfect, $D(S_0)$ can differ from Y_0 even when $\alpha = 0$. We therefore retain the reconstruction residual

$$R_0 = Y_0 - D(S_0). \tag{5}$$

It captures the part of the base forecast not recovered by the autoencoder. Adding R_0 to $D(S_{\mathrm{ref}})$ yields $Y_0 + [D(S_{\mathrm{ref}}) - D(S_0)]$, which exactly recovers Y_0 when $\alpha = 0$.

To adapt the contribution of this residual, we introduce another learnable scalar β, shared across all inputs. Using $Y_0 = D(S_0) + R_0$, the final forecast is

$$Y_{\mathrm{ref}} = \underbrace{Y_0}_{\text{Forecast}} + \underbrace{D\big(S_0 + \alpha \underbrace{(\widehat{S} - S_0)}_{\text{Guide}}\big) - D(S_0) + \beta R_0}_{\text{Refine}}. \tag{6}$$

Thus, refinement preserves the base forecast while adding a decoded state correction and an adaptive residual adjustment.

3.5 TRAINING OBJECTIVE

Before training FUSE, we pretrain the autoencoder on training windows V using $\mathcal{L}_{\mathrm{AE}} = \mathrm{MAE}(D(E(V)), V)$. The base forecaster F is trained independently on the forecasting task. After this preparation, F, E, and D remain frozen during FUSE training.

State prediction loss. We supervise the predicted state $\widehat{S}$ with the encoded future $S^\star = E(Y)$. The State prediction loss combines the State MSE loss and State cosine loss (Yang et al., 2026):

$$\mathcal{L}_{\mathrm{mse}} = \frac{1}{Hd}\|\widehat{S} - S^\star\|_F^2, \qquad \mathcal{L}_{\cos} = \frac{1}{H}\sum_{t=1}^{H}\left[1 - \cos(\widehat{s}_t, s_t^\star)\right]. \tag{7}$$

where $\widehat{s}_t, s_t^\star \in \mathbb{R}^d$ are the state vectors at future time t. State MSE penalizes numerical discrepancies, while State cosine loss adds angular supervision at each time step. Since the predicted and target displacements share the starting state S_0, State MSE also measures their displacement error (Appendix B.1).

Forecasting loss. To optimize the resulting forecast, we directly supervise Y_{ref} in the observation space:

$$\mathcal{L}_{\mathrm{forecast}} = \mathrm{MSE}(Y_{\mathrm{ref}}, Y) = \frac{1}{HC}\|Y_{\mathrm{ref}} - Y\|_F^2. \tag{8}$$

It trains the overall output change to compensate for the base forecasting error, as shown in Appendix B.2.

Joint objective and optimization. The joint training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{forecast}} + \lambda_{\mathrm{m}}\mathcal{L}_{\mathrm{mse}} + \lambda_{\mathrm{c}}\mathcal{L}_{\cos}, \tag{9}$$

where λ_{m} and λ_{c} are fixed loss weights. We optimize the future state predictor P_θ together with the coefficients α and β. Both coefficients are initialized to zero, so FUSE initially reproduces the base forecast Y_0. State supervision provides a direct training signal to the predictor at this initialization. The interpretation of the joint objective and its gradient paths are derived in Appendix B.3.

Table 1: Long-term multivariate forecasting results averaged over four horizons $H \in \{96, 192, 336, 720\}$. For each method, dataset, and horizon, the lookback length is selected from $L \in \{336, 512, 720\}$. The best and second-best results are highlighted in **bold** and underline. Full results are provided in Appendix D.1.

Models	**FUSE (Ours)**		TimeAlign (2026)		CMoS (2025)		TimeBase (2025)		PDF (2024)		iTransformer (2024)		TimeMixer (2024a)		PatchTST (2023)		TimesNet (2023)		DLinear (2023)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.334**	**0.366**	0.340	0.367	0.354	0.379	0.357	0.381	0.342	0.376	0.362	0.391	0.356	0.380	0.353	0.376	0.399	0.406	0.356	0.378
ETTm2	**0.245**	**0.304**	0.246	0.306	0.259	0.317	0.251	0.314	0.250	0.313	0.269	0.329	0.257	0.318	0.256	0.317	0.292	0.331	0.259	0.324
ETTh1	0.398	0.418	0.406	0.420	0.403	0.416	**0.396**	**0.415**	0.407	0.426	0.439	0.448	0.427	0.441	0.419	0.437	0.488	0.452	0.425	0.439
ETTh2	0.337	**0.382**	0.338	**0.382**	**0.331**	0.383	0.347	0.398	0.347	0.391	0.374	0.406	0.349	0.397	0.351	0.395	0.390	0.417	0.431	0.447
Weather	**0.217**	0.253	0.219	**0.248**	0.220	0.261	0.219	0.263	0.227	0.263	0.233	0.271	0.226	0.264	0.226	0.264	0.259	0.287	0.242	0.293
Electricity	**0.152**	**0.244**	0.155	0.245	0.158	0.250	0.167	0.258	0.160	0.253	0.164	0.261	0.173	0.272	0.159	0.253	0.193	0.295	0.166	0.264
Traffic	**0.389**	0.255	**0.389**	**0.249**	0.396	0.271	0.418	0.279	0.395	0.270	0.397	0.281	0.409	0.279	0.397	0.275	0.620	0.336	0.418	0.287
Solar	**0.186**	**0.217**	0.195	0.220	0.210	0.257	0.237	0.271	0.200	0.263	0.202	0.248	0.193	0.252	0.194	0.245	0.244	0.334	0.223	0.226

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate FUSE on eight multivariate forecasting benchmarks: the four ETT subsets (ETTh1, ETTh2, ETTm1, ETTm2) (Zhou et al., 2021), Weather, Electricity, Traffic (Wu et al., 2023), and Solar (Liu et al., 2024). More details about datasets are described in Appendix C.1.

Baselines. We compare FUSE with nine forecasting methods: TimeAlign (Hu et al., 2026), PDF (Dai et al., 2024), CMoS (Si et al., 2025), TimeBase (Huang et al., 2025), iTransformer (Liu et al., 2024), PatchTST (Nie et al., 2023), TimeMixer (Wang et al., 2024a), TimesNet (Wu et al., 2023), and DLinear (Zeng et al., 2023). Baseline details are provided in Appendix C.2.

Implementation Details. To keep consistent with previous works, we report mean squared error (MSE) and mean absolute error (MAE) for forecast horizons $H \in \{96, 192, 336, 720\}$. For performance comparisons, we conduct a grid search on the lookback lengths of $L \in \{336, 512, 720\}$ and report each method's best results for fair comparisons, following the lookback-search practice in prior MTSF studies (Qiu et al., 2025; Hu et al., 2026). For each configuration, all reported FUSE results are averaged over three runs with different random seeds. FUSE is implemented in PyTorch (Paszke et al., 2019) and trained on one NVIDIA RTX 4090 24 GB GPU. Further evaluation and training details are given in Appendices C.2 and C.3.

4.2 PERFORMANCE EVALUATION

Forecasting performance. Table 1 summarizes forecasting results averaged over the four horizons. FUSE ranks first or second in 15 of the 16 comparisons across datasets and metrics, including ties. This demonstrates consistently strong performance across the benchmark suite. It achieves the best MSE and MAE on ETTm1, ETTm2, Electricity, and Solar. On Solar, it reduces MSE by 3.63% relative to TimeMixer and MAE by 1.36% relative to TimeAlign, the strongest baselines for the respective metrics. Overall, the complete FUSE framework achieves leading accuracy across multiple benchmarks by combining a simple MLP base forecaster with future state guidance and forecast refinement.

Plug-in performance. We evaluate FUSE as a plug-in for the default MLP, DLinear (Zeng et al., 2023), and iTransformer (Liu et al., 2024), with all pretrained backbones frozen. Following Section 4.1, Table 2 reports results averaged over four forecasting horizons. FUSE reduces MSE and achieves positive Gain in all 24 dataset–backbone combinations, with average gains of 3.64%, 4.69%, and 4.14% for the three backbones, respectively. The only MAE deterioration occurs for DLinear on Solar, where MAE increases by 4.03% while MSE decreases by 13.53%. Consistent with the independent state-geometry diagnostic in Appendix A.2, Solar contains a larger fraction of favorable predicted endpoints, suggesting that FUSE can benefit both by tempering imperfect

Table 2: Plug-in performance averaged over horizons $H \in \{96, 192, 336, 720\}$. **Bold** indicates improvement and gray indicates degradation relative to Base. Gain (%) averages the relative MSE and MAE reductions; positive gains are shown in **bold**. Full results are provided in Appendix D.2.

Dataset	Default MLP					DLinear					iTransformer				
	Base		+FUSE		Gain (%)	Base		+FUSE		Gain (%)	Base		+FUSE		Gain (%)
	MSE	MAE	MSE	MAE		MSE	MAE	MSE	MAE		MSE	MAE	MSE	MAE	
ETTm1	0.351	0.368	**0.334**	**0.366**	**+2.78**	0.356	0.378	**0.341**	**0.373**	**+2.68**	0.362	0.391	**0.342**	**0.375**	**+4.87**
ETTm2	0.264	0.312	**0.245**	**0.304**	**+4.71**	0.259	0.324	**0.248**	**0.313**	**+3.82**	0.269	0.329	**0.252**	**0.313**	**+5.58**
ETTh1	0.415	0.426	**0.398**	**0.418**	**+3.07**	0.425	0.439	**0.417**	**0.436**	**+1.23**	0.439	0.448	**0.422**	**0.443**	**+2.52**
ETTh2	0.342	0.384	**0.337**	**0.382**	**+1.04**	0.431	0.447	**0.369**	**0.408**	**+11.55**	0.374	0.406	**0.354**	**0.392**	**+4.28**
Weather	0.231	0.253	**0.217**	**0.253**	**+3.08**	0.242	0.293	**0.225**	**0.265**	**+8.13**	0.233	0.271	**0.225**	**0.264**	**+3.11**
Electricity	0.165	0.249	**0.152**	**0.244**	**+4.98**	0.166	0.264	**0.157**	**0.257**	**+3.87**	0.164	0.261	**0.153**	**0.250**	**+5.37**
Traffic	0.401	0.262	**0.389**	**0.255**	**+3.04**	0.418	0.287	**0.415**	**0.281**	**+1.49**	0.397	0.281	**0.394**	**0.277**	**+1.07**
Solar	0.209	0.221	**0.186**	**0.217**	**+6.46**	0.223	0.226	**0.193**	0.235	**+4.75**	0.202	0.248	**0.189**	**0.232**	**+6.30**

Table 3: Contributions to MSE reduction (%), averaged over $H \in \{96, 192, 336, 720\}$ with $L = 336$. Mean, Scaling, and Shape denote mean shift, waveform scaling, and residual shape correction. Comparisons are made before rounding. Full results are provided in Appendix D.3.5

Dataset	Default MLP			DLinear			iTransformer		
	Mean	Scaling	Shape	Mean	Scaling	Shape	Mean	Scaling	Shape
ETTm1	**+1.22**	**+1.03**	**+0.42**	**+1.12**	**+2.02**	**+0.34**	**+3.57**	**+0.93**	**+2.93**
ETTm2	**+0.91**	**+0.27**	**+0.48**	**+8.50**	**+0.90**	**+0.93**	**+2.10**	**+1.23**	**+3.05**
ETTh1	**+0.30**	**+0.93**	**+0.62**	**+0.05**	**+0.01**	**+0.00**	**+1.66**	**+0.96**	**+3.35**
ETTh2	**+0.97**	**+0.34**	**+0.92**	**+15.20**	**+2.39**	**+0.93**	**+3.46**	**+0.76**	**+2.64**
Weather	**+0.45**	**+0.54**	**+0.15**	**+6.07**	**+1.29**	**+0.12**	**+3.60**	**+1.30**	**+0.49**
Electricity	**+3.18**	**+0.52**	**+0.38**	**+0.55**	**+0.16**	**+1.56**	**+2.43**	**+0.57**	**+1.69**
Traffic	**+1.11**	**+2.36**	-0.02	-4.66	**+4.81**	**+0.42**	**+3.09**	-5.23	**+0.85**
Solar	**+1.14**	**+1.33**	**+0.52**	**+4.09**	**+7.62**	**+5.12**	**+1.76**	**+1.07**	**+2.19**

displacements and by retaining or further exploiting useful ones. Overall, future state guidance generalizes across pretrained forecasting models and enables plug-in refinement directly from their outputs, without modifying or jointly retraining the backbone.

4.3 Analysis of Forecast Refinement

We examine how FUSE changes a base forecast by decomposing the complete output change $Y_{\text{ref}} - Y_0$ into three components: **Mean Shift**, **Waveform Scaling**, and **Residual Shape Correction**. This change includes both decoded state refinement and reconstruction residual adjustment. Mean Shift uniformly raises or lowers the prediction level without changing differences between time steps. Waveform Scaling proportionally amplifies or attenuates fluctuations along the centered base waveform. Residual Shape Correction changes temporal patterns beyond uniform shifting and scaling, such as adjustments to individual peaks and troughs. We quantify each component's contribution by its reduction in squared prediction error, normalized by the base forecast's total MSE. Table 3 reports these contributions averaged equally over the four horizons. Appendix D.3 provides the formal definitions and aggregation protocol.

Table 3 shows that all three components contribute to error reduction in most dataset and backbone combinations. On Electricity, which exhibits strong seasonality (Qiu et al., 2024), Mean Shift is the largest contributor for the default MLP at 3.18%. This shows that strong recurring patterns can coexist with room for improving the predicted level of a future window. On Solar, characterized by high stationarity (Qiu et al., 2024), Waveform Scaling contributes 7.62% for DLinear. The improvement mainly comes from adjusting the amplitude of predicted fluctuations, as stable data statistics do not necessarily prevent amplitude errors in the base forecast that can be corrected through waveform rescaling. In addition to level and amplitude, Residual Shape Correction is the largest contributor for iTransformer on ETTm2, at 3.05%. This indicates that FUSE can further correct temporal variations that cannot be captured by uniform shifts or rescaling alone. Together, these results show that FUSE refines the initial forecast at multiple levels, correcting its overall level, fluctuation amplitude, and local temporal patterns. On Traffic, however, Mean Shift contributes negatively for DLinear,

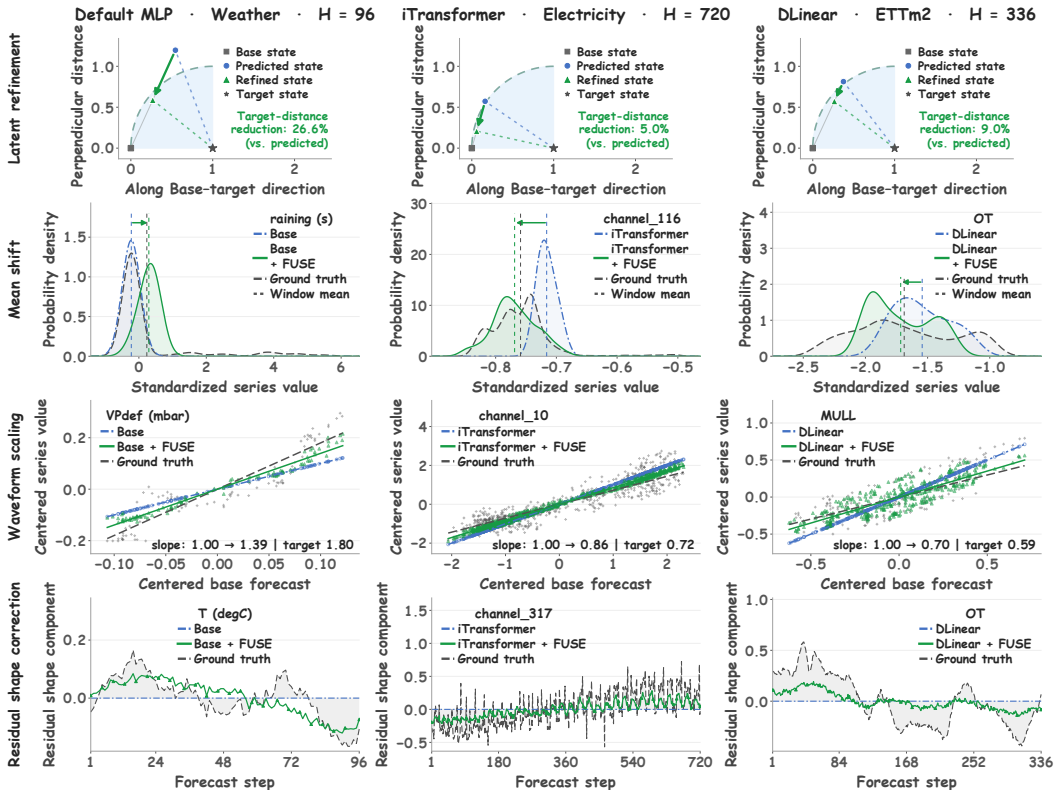

Figure 4: **Examples of state refinement and the corresponding forecast changes**. Each column uses one multivariate test window at $L = 336$. State coordinates are normalized by $\|S^\star - S_0\|_F$.

whereas Waveform Scaling does so for iTransformer. This indicates that the benefits of individual refinement components depend on the backbone and are not uniformly positive across settings.

Figure 4 presents representative cases for the default MLP on Weather, iTransformer on Electricity, and DLinear on ETTm2. Each column corresponds to one multivariate forecasting window, with displayed channels identified in the legends. In the top row, the predicted future states lie outside the reference boundaries and are farther from the target states than corresponding base states. By adjusting the displacement from the base states, FUSE moves the refined states toward the target and back within the reference boundaries. The remaining rows visualize how this state-level guidance translates into different corrections to the final forecast. In the mean plots, the green refined mean lies closer to the gray target mean than the blue base mean, showing that FUSE corrects over- or underestimation of the average level over the forecast horizon. In the scaling plots, each displayed point pairs a centered series value with the centered base forecast at the same time step. The fitted slope after refinement moves closer to the target slope than original unit slope, indicating FUSE can amplify underestimated fluctuations or suppress excessive ones while preserving the underlying base waveform. In the shape plots, the base component is zero by construction after removing both its mean and the component aligned with the base waveform. The refined green curves recover part of the remaining temporal structure in the gray target, thereby reducing the residual shape error relative to the baseline. These examples illustrate how future state guidance translates into complementary corrections of the level, fluctuation scale, and residual temporal shape of an existing forecast.

4.4 Ablation Study

We evaluate four FUSE variants using the default MLP backbone. Specifically, $\alpha = 0$ disables state refinement and $\alpha = 1$ uses the predicted state directly, $S_{\text{ref}} = \widehat{S}$, both with learnable β. Conversely, $\beta = 0$ keeps reconstruction residual R_0 unchanged and $\beta = -1$ removes it, both with learnable α. As shown in Table 4, FUSE achieves the lowest MSE and MAE across datasets. Without state

Table 4: Component ablations with the default MLP, averaged over four horizons. Each ablated variant fixes the indicated coefficient during training. Best results are in **bold**.

Dataset	Base		$\alpha = 0$, β learned		$\alpha = 1$, β learned		$\beta = 0$, α learned		$\beta = -1$, α learned		**Full FUSE**	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	0.351	0.368	0.351	0.368	0.349	0.378	0.342	0.371	0.342	0.371	**0.334**	**0.366**
ETTm2	0.264	0.312	0.264	0.313	0.262	0.316	0.260	0.310	0.260	0.311	**0.245**	**0.304**
ETTh1	0.415	0.426	0.415	0.426	0.411	0.432	0.409	0.425	0.409	0.425	**0.398**	**0.418**
ETTh2	0.342	0.384	0.339	0.384	0.338	0.383	0.342	0.388	0.339	0.386	**0.337**	**0.382**
Weather	0.231	0.253	0.229	0.262	0.234	0.269	0.230	0.261	0.228	0.259	**0.217**	**0.253**
Electricity	0.165	0.249	0.164	0.249	0.165	0.264	0.158	0.248	0.162	0.258	**0.152**	**0.244**
Traffic	0.401	0.262	0.414	0.257	0.468	0.312	0.401	0.262	0.556	0.310	**0.389**	**0.255**
Solar	0.209	0.221	0.209	0.218	0.204	0.239	0.201	0.232	0.201	0.229	**0.186**	**0.217**

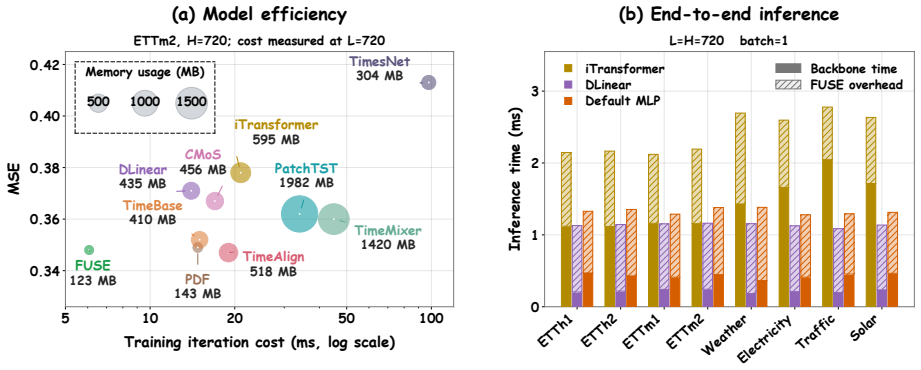

Figure 5: **Computational efficiency**. (a) Accuracy and training cost on ETTm2 dataset. FUSE's reported cost excludes AE pretraining. (b) End-to-end inference time with batch size one.

refinement, performance is close to Base, showing residual adjustment alone provides limited gains. Using the predicted state as the refinement target degrades performance on Weather, Electricity, and Traffic, supporting the need to learn the state displacement. This suggests the predicted future state is informative but not sufficiently accurate as a fixed target. Fixing $\beta = 0$ also reduces accuracy, while removing R_0 causes larger degradation on Traffic, indicating that the residual preserves forecast information not fully represented in the decoded state. These results confirm the complementary roles of learned state refinement and adaptive residual adjustment. Additional analyses of the state losses, loss-weight sensitivity, and learned state-refinement step are provided in Appendix E.

4.5 Computational Efficiency

Figure 5 summarizes accuracy and efficiency. On ETTm2, FUSE achieves competitive accuracy with 123 MB peak memory, while one base and FUSE training iteration require 6.07 ms total. Across datasets and backbones, FUSE adds only 0.73–1.26 ms at batch size one, with total inference times of 1.09–2.78 ms. Overall, FUSE maintains competitive accuracy with millisecond-level overhead.

5 Conclusion

We identify *Predicted State Deviation*, where future states learned under supervision from encoded targets can be less accurate than the corresponding base states. FUSE treats these imperfect states as guidance and learns how far to move along or against their displacement from the base state. It then combines the decoded refined state with an adjusted reconstruction residual to preserve useful information from the base forecast. Experiments on eight datasets demonstrate competitive accuracy with a simple MLP and consistent average improvements across three backbones as a plug-in, without modifying the underlying backbone. Ablation studies validate the refinement design, while decomposition analyses show how FUSE corrects the forecast mean, waveform scale, and residual shape. More broadly, this study suggests that the value of future state prediction may lie not only in its accuracy, but in how its imperfect information can be leveraged to refine existing forecasts.

AI USE STATEMENT

Generative AI tools were utilized to assist in the writing process, specifically to refine language and improve readability. These tools did not influence the research design, experimental implementation, or analysis. All methodological contributions, experiments, and conclusions were conceived and executed independently.

ETHICS STATEMENT

This study focuses on multivariate time series forecasting problem and uses public time series benchmarks. It involves no human-subject experiments or collection of new personal data. Therefore, there is no potential ethical risks.

REPRODUCIBILITY STATEMENT

Section 3 specifies the architecture, refinement mechanism, and training objectives. Appendix C documents the datasets, preprocessing, evaluation protocol, and implementation settings. Appendices B and D.3 provide the mathematical derivations and forecast decomposition definitions. Horizon-wise forecasting and plug-in results are reported in Appendix D.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Assran_Self-Supervised_Learning_From_Images_With_a_Joint-Embedding_Predictive_Architecture_CVPR_2023_paper.html.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1298–1312. PMLR, 2022. URL https://proceedings.mlr.press/v162/baevski22a.html.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 9780521833783. URL https://web.stanford.edu/~boyd/cvxbook/.

Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/2be6705de7412adf107900add727a795-Abstract-Conference.html.

Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Jigang Bao, Yong Jiang, and Shu-Tao Xia. Periodicity decoupling framework for long-term series forecasting. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/d298cf34e4539f9134db7f38b42f69fe-Abstract-Conference.html.

Sungwon Han, Seungeon Lee, Meeyoung Cha, Sercan O. Arik, and Jinsung Yoon. Retrieval augmented time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21774–21797. PMLR, 2025. URL https://proceedings.mlr.press/v267/han25d.html.

Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In *Computer Vision – ECCV 2020*, pp. 312–329. Springer, 2020. doi: 10.1007/978-3-030-58580-8_19. URL https://doi.org/10.1007/978-3-030-58580-8_19.

Yifan Hu, Jie Yang, Tian Zhou, Peiyuan Liu, Yujin Tang, Rong Jin, and Liang Sun. Bridging past and future: Distribution-aware alignment for time series forecasting. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pQzQfslqlD.

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 26227–26246. PMLR, 2025. URL https://proceedings.mlr.press/v267/huang25az.html.

Jeonghwan Im and Hyuk-Yoon Kwon. COSA: Context-aware output-space adapter for test-time adaptation in time series forecasting. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/2a8ce71baac4c89bf9ff479d8240c7d9-Abstract-Conference.html.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

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 *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/6f7d90b1198fec96defd80b5ebd5bc81-Abstract-Conference.html.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379 (2194):20200209, 2021. doi: 10.1098/rsta.2020.0209. URL https://doi.org/10.1098/rsta.2020.0209.

Peiyuan Liu, Beiliang Wu, Yifan Hu, Naiqi Li, Tao Dai, Jigang Bao, and Shu-Tao Xia. TimeBridge: Non-stationarity matters for long-term time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 39815–39840. PMLR, 2025a. URL https://proceedings.mlr.press/v267/liu25cb.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, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/28b3dc0970fa4624a63278a4268de997-Abstract-Conference.html.

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*, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

Zhiding Liu, Mingyue Cheng, Guanhao Zhao, Jiqian Yang, Qi Liu, and Enhong Chen. Improving time series forecasting via instance-aware post-hoc revision. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. URL https://papers.neurips.cc/paper_files/paper/2025/hash/331c41353b053683e17f7c88a797701d-Abstract-Conference.html.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1608.03983.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Donghao Luo and Xue Wang. ModernTCN: A modern pure convolution structure for general time series analysis. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vpJMJerXHU.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. TFB: Towards comprehensive and fair benchmarking of time series forecasting methods. *Proceedings of the VLDB Endowment*, 17 (9):2363–2377, 2024. doi: 10.14778/3665844.3665863. URL https://www.vldb.org/pvldb/vol17/p2363-hu.pdf.

Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. DUET: Dual clustering enhanced multivariate time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 1185–1196. Association for Computing Machinery, 2025. doi: 10.1145/3690624.3709325. URL https://doi.org/10.1145/3690624.3709325.

Dmitriy Serdyuk, Nan Rosemary Ke, Alessandro Sordoni, Adam Trischler, Chris Pal, and Yoshua Bengio. Twin networks: Matching the future for sequence generation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BydLzGb0Z.

Haotian Si, Changhua Pei, Jianhui Li, Dan Pei, and Gaogang Xie. CMoS: Rethinking time series prediction through the lens of chunk-wise spatial correlations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55493–55510. PMLR, 2025. URL https://proceedings.mlr.press/v267/si25a.html.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. URL https://arxiv.org/abs/1807.03748.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/d54e99a6c03704e95e6965532dec148b-Paper.pdf.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Zhang, and Jun Zhou. TimeMixer: Decomposable multiscale mixing for time series forecasting. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/a7ac8a21e5a27e7ab31a5f42a0117bdb-Abstract-Conference.html.

Xue Wang, Tian Zhou, Qingsong Wen, Jinyang Gao, Bolin Ding, and Rong Jin. CARD: Channel aligned robust blend transformer for time series forecasting. In *International Conference on Learning Representations*, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/2f4d6f8e0f4f543db12260696b2a3551-Abstract-Conference.html.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2D-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Jie Yang, Yifan Hu, Yuante Li, Kexin Zhang, Kaize Ding, and Philip S. Yu. From observations to states: Latent time series forecasting. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=s49fw3BVU0.

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. doi: 10.1609/aaai.v37i9.26317. URL https://ojs.aaai.org/index.php/AAAI/article/view/26317.

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. doi: 10.1609/aaai.v35i12.17325. URL https://ojs.aaai.org/index.php/AAAI/article/view/17325.

A State Geometry and Diagnostic Analysis

This section formalizes the geometry underlying Figure 2 and specifies the diagnostic protocol.

A.1 Geometry of Predicted State Deviation

For test window i, let $S_{0,i}$, $\widehat{S}_i$, and $S_i^\star$ denote the base, predicted, and target states. Define

$$t_i = \mathrm{vec}(S_i^\star - S_{0,i}), \qquad d_i = \mathrm{vec}(\widehat{S}_i - S_{0,i}), \qquad r_i = \|t_i\|_2. \tag{10}$$

Vectorization preserves Frobenius norms. For $r_i > 0$, we define the plotting coordinates

$$x_i = \frac{\langle d_i, t_i \rangle}{r_i^2}, \qquad w_i = d_i - x_i t_i, \qquad y_i = \frac{\|w_i\|_2}{r_i}. \tag{11}$$

By construction,

$$\langle w_i, t_i \rangle = \langle d_i, t_i \rangle - x_i r_i^2 = 0. \tag{12}$$

Thus, x_i measures displacement along the target direction, while $y_i \geq 0$ measures the perpendicular deviation. Under this normalization, the base and target states are mapped to $(0, 0)$ and $(1, 0)$, respectively.

Using $d_i = x_i t_i + w_i$ and orthogonality,

$$\begin{aligned} \|d_i\|_2^2 &= x_i^2 r_i^2 + \|w_i\|_2^2 = r_i^2 (x_i^2 + y_i^2), \\ \|d_i - t_i\|_2^2 &= (x_i - 1)^2 r_i^2 + \|w_i\|_2^2 = r_i^2 \left[(x_i - 1)^2 + y_i^2 \right]. \end{aligned} \tag{13}$$

The plot therefore preserves the normalized distances from each predicted state to its own base and target states. The equal-distance boundary is

$$(x - 1)^2 + y^2 = 1. \tag{14}$$

Predicted states outside this boundary are farther from the target than their corresponding base states, providing a geometric characterization of *Predicted State Deviation*. Because each window uses its own target direction and normalization, distances between different windows are not preserved.

The predicted endpoint, however, is only one possible use of its displacement. To analyze the available movement, keep the three states fixed and consider

$$S_i(a) = S_{0,i} + a(\widehat{S}_i - S_{0,i}), \qquad a \in \mathbb{R}. \tag{15}$$

Here, $a = 0$ corresponds to the base state, $a = 1$ reaches the predicted state, $0 < a < 1$ interpolates between them, $a > 1$ extrapolates beyond the predicted state, and $a < 0$ reverses the predicted displacement.

The normalized squared distance to the target is

$$\begin{aligned} \rho_i(a) &= \frac{\|S_i(a) - S_i^\star\|_F^2}{r_i^2} = \frac{\|a d_i - t_i\|_2^2}{r_i^2} \\ &= (a x_i - 1)^2 + a^2 y_i^2 \\ &= 1 - 2a x_i + a^2 (x_i^2 + y_i^2). \end{aligned} \tag{16}$$

The base state satisfies $\rho_i(0) = 1$, while $\rho_i(1)$ measures the relative quality of the predicted endpoint.

For $d_i \neq 0$, completing the square gives

$$\begin{aligned} \rho_i(a) &= (x_i^2 + y_i^2) \left(a - a_i^{\mathrm{geom}}\right)^2 + 1 - \frac{x_i^2}{x_i^2 + y_i^2}, \\ a_i^{\mathrm{geom}} &= \frac{x_i}{x_i^2 + y_i^2} = \frac{\langle d_i, t_i \rangle}{\|d_i\|_2^2}. \end{aligned} \tag{17}$$

Since $x_i^2 + y_i^2 > 0$, a_i^{geom} is the unique step that minimizes target distance along the predicted displacement. This distinguishes endpoint accuracy from directional usefulness: an inaccurate predicted state can still define a direction containing a better intermediate or reversed state.

Moreover,

$$\rho_i(a) - 1 = (x_i^2 + y_i^2)a\,(a - 2a_i^{\text{geom}})\,. \tag{18}$$

Hence, a step improves upon the base state exactly when

$$0 < a < 2a_i^{\text{geom}} \tag{19}$$

for $x_i > 0$, or

$$2a_i^{\text{geom}} < a < 0 \tag{20}$$

for $x_i < 0$.

When

$$0 < a_i^{\text{geom}} < 1, \tag{21}$$

the closest point lies between the base and predicted states, so interpolation is preferable to using the predicted endpoint directly. In particular, if the endpoint is worse than the base,

$$\rho_i(1) > 1 \quad \Longleftrightarrow \quad x_i^2 + y_i^2 > 2x_i. \tag{22}$$

Together with $x_i > 0$, this implies

$$0 < a_i^{\text{geom}} < \frac{1}{2}. \tag{23}$$

Thus, even when the full displacement worsens the state, a smaller movement along the same direction can still improve it. This is the key geometric case underlying the use of imperfect predicted states as guidance rather than fixed refinement targets.

When

$$a_i^{\text{geom}} > 1, \tag{24}$$

the closest point lies beyond the predicted endpoint, so extrapolation is beneficial. In this case,

$$0 < x_i^2 + y_i^2 < x_i \tag{25}$$

implies $\rho_i(1) < 1$, and

$$\rho_i(1) - \rho_i(a_i^{\text{geom}}) = (x_i^2 + y_i^2)\,(1 - a_i^{\text{geom}})^2 > 0. \tag{26}$$

Hence, an endpoint that already improves upon the base can still be further improved by moving farther along the same displacement.

When

$$a_i^{\text{geom}} < 0, \tag{27}$$

the predicted displacement has a negative inner product with the target direction. Positive steps therefore increase target distance, whereas reversing the displacement improves the state whenever

$$2a_i^{\text{geom}} < a < 0. \tag{28}$$

This provides the geometric interpretation of moving *against* a predicted displacement.

The remaining cases follow directly. At $a_i^{\text{geom}} = 1$, the predicted endpoint is optimal along the line. If $x_i = 0$ and $d_i \neq 0$, every nonzero step increases target distance. If $d_i = 0$, all values of a leave the state unchanged. If $r_i = 0$, the base state already equals the target, so the normalized coordinates are undefined and no strict improvement is possible.

For comparison, the diagnostic also computes the best step restricted to the segment between the base and predicted states:

$$a_i^{\text{seg}} = \min\{1, \max\{0, a_i^{\text{geom}}\}\}\,. \tag{29}$$

This is obtained by minimizing Equation 17 over $a \in [0, 1]$; we set $a_i^{\text{seg}} = 0$ when $d_i = 0$.

Relation to a shared refinement coefficient. Both a_i^{geom} and a_i^{seg} depend on the true target and are used only for diagnosis. They characterize the best movement for each window, whereas FUSE learns one coefficient shared across inputs.

To examine this distinction, consider a fixed collection of N windows with $r_i > 0$ and keep their states unchanged. Let $q_i = x_i^2 + y_i^2$. Applying the same coefficient a to every window gives the average normalized squared target distance

$$\begin{aligned} \bar{\rho}(a) &= \frac{1}{N}\sum_{i=1}^{N}\rho_i(a) = 1 - 2a\bar{x} + a^2\bar{q}, \\ \bar{x} &= \frac{1}{N}\sum_{i=1}^{N}x_i, \qquad \bar{q} = \frac{1}{N}\sum_{i=1}^{N}q_i. \end{aligned} \tag{30}$$

For $\bar{q} > 0$, completing the square yields

$$\begin{aligned} \bar{\rho}(a) &= 1 - \frac{\bar{x}^2}{\bar{q}} + \bar{q}\left(a - \frac{\bar{x}}{\bar{q}}\right)^2, \\ a_{\mathrm{shared}}^{\mathrm{geom}} &= \frac{\bar{x}}{\bar{q}} = \frac{\sum_{i:q_i>0} q_i a_i^{\mathrm{geom}}}{\sum_{i=1}^{N} q_i}. \end{aligned} \tag{31}$$

Windows with $q_i = 0$ have zero displacement and contribute only a constant to the objective. The shared geometric optimum is therefore a weighted average of the window-wise optima. It need not minimize each window's distance, and beneficial window-wise oracle movements do not guarantee improvement from a common nonzero coefficient.

This geometric optimum is distinct from the coefficient learned by FUSE. FUSE optimizes α jointly with the future state predictor and β through the final forecasting loss, which includes decoding and reconstruction residual adjustment (Appendix D.3.3). At inference, α is fixed and requires no target information. Appendix E.2 evaluates its actual forecasting benefit by comparing the learned shared coefficient with reaching the same predicted endpoint.

A.2 Diagnostic Protocol and Full Results

We train independent DLinear and iTransformer state predictors (Zeng et al., 2023; Liu et al., 2024) on eight datasets with $L = 336$, $H \in \{96, 336\}$. These settings follow the data splits and standardization described in Appendix C.1. Each setting also uses a separately trained observation forecaster. A frozen dataset-specific encoder defines

$$S_0 = E(Y_0), \qquad S^\star = E(Y), \tag{32}$$

and is shared across predictor families and horizons.

The diagnostic predictors receive historical observations, whereas the FUSE predictor operates on encoded history. We retain each temporal backbone and append a linear projection from C variables to d state dimensions at each future step. For iTransformer, output denormalization is omitted before this projection. Writing the independent predictor as $\widetilde{P}_\phi$, we optimize

$$\mathcal{L}_{\mathrm{diag}} = \frac{1}{Hd}\left\|\widetilde{P}_\phi(X) - E(Y)\right\|_F^2. \tag{33}$$

Only ϕ is updated, using state MSE without cosine or decoded forecasting losses. Historical time features are provided where required by the backbone. True future observations are used only for supervision and geometric evaluation, never as predictor inputs.

DLinear uses a moving-average kernel of 25 and shared linear mappings across variables. iTransformer uses eight heads, GELU activations, and dropout of 0.1. Its embedding and feed-forward dimensions are 128 on ETT datasets and 512 elsewhere. It uses two layers on ETT, four on Traffic, and three on the remaining datasets. State dimensions and autoencoder training follow Appendix C.3.

Training uses AdamW (Loshchilov & Hutter, 2019), cosine learning-rate decay (Loshchilov & Hutter, 2017), weight decay 10^{-4}, and gradient-norm clipping at 1.0. The maximum budget is

50 epochs, with a minimum of five epochs, patience eight, and a relative improvement threshold of 10^{-4}. Checkpoints are selected by validation state MSE. DLinear uses a learning rate of 10^{-4} throughout. iTransformer uses 10^{-4} except on Electricity and Traffic, where the rates are 5×10^{-4} and 10^{-3}, respectively. For ETT, Weather, Electricity, Traffic, and Solar, the batch sizes are $(512, 512, 128, 32, 128)$ for DLinear and $(256, 256, 32, 16, 64)$ for iTransformer.

For N test windows, we report

$$
\begin{aligned}
p_{+} &= \frac{100}{N} \sum_{i} \mathbf{1}[x_i > 0], \\
p_{\mathrm{w}} &= \frac{100}{N} \sum_{i} \mathbf{1}[\rho_i(1) > 1], \\
p_{\mathrm{ws}} &= \frac{100}{N} \sum_{i} \mathbf{1}[x_i > 0,\ \rho_i(1) > 1,\ \rho_i(a_i^{\mathrm{seg}}) < 1].
\end{aligned}
\tag{34}
$$

Here, p_{+} measures the fraction of windows whose predicted displacement has a positive projection onto the target direction. The quantity p_{w} measures how often the predicted endpoint is farther from the target than the base state, directly quantifying *Predicted State Deviation*. Most importantly, p_{ws} measures windows in which the predicted endpoint is worse than the base, yet a partial movement along the same displacement yields a better state. Thus, p_{ws} directly quantifies the "worse endpoint, useful direction" phenomenon established geometrically in Appendix A.1. Commas in the final indicator denote conjunction.

We also report

$$
\bar{\rho}_{\mathrm{end}} = \frac{1}{N} \sum_{i} \rho_i(1), \qquad \bar{\rho}_{\mathrm{seg}} = \frac{1}{N} \sum_{i} \rho_i(a_i^{\mathrm{seg}}),
\tag{35}
$$

which average the normalized squared target distances of the predicted endpoint and the best segment point, respectively. Because normalization is performed within each window, the base state has normalized distance one. Squared-norm denominators are floored at 10^{-12}, negative roundoff in y_i^2 is clipped to zero, and all indicators are computed from unrounded values.

Figure 2 visualizes seven datasets using all test windows, with centroids computed from the complete point clouds and equal horizontal and vertical scales in each panel. Table 5 reports the complete scan, including Solar.

Predicted State Deviation is widespread. In 28 of the 32 settings, more than 80% of test windows have a predicted endpoint farther from the target than the corresponding base state. However, endpoint deterioration does not imply that the predicted displacement is uninformative. In 17 settings, more than half of the windows combine a worse endpoint with a better partial state, and the mean p_{ws} across all settings is 54.75%. Thus, in a substantial fraction of windows, the predicted state is inaccurate as an endpoint while still providing a direction that contains an improved intermediate state.

Solar exhibits a different geometry, with p_{w} ranging from 11.48% to 57.52%, indicating a substantially larger fraction of predicted endpoints that already improve upon the base state. This does not contradict the refinement principle: useful state guidance does not require endpoint deterioration. When the predicted endpoint is already favorable, refinement can retain or further exploit its displacement rather than correcting it back toward the base state. The segment statistics reported here evaluate interpolation only; reversal and extrapolation are characterized by the unrestricted geometry in Appendix A.1.

A.3 Geometry of Trained FUSE Predictors

We further examine whether Predicted State Deviation persists in the future state predictors after joint FUSE training. We evaluate all eight datasets and $H \in \{96, 192, 336, 720\}$, retaining the selected lookback length for each setting. With all parameters frozen, we obtain the encoded base forecast, predicted future state, and encoded target for each test window, using true futures only for this post-hoc diagnostic. We apply the same geometric definitions as in Appendix A.2, compute the statistics within each checkpoint, and average them equally across seeds and horizons.

Table 5: State-geometry diagnostics at $L = 336$. N counts test windows; the three p columns are percentages. The $\bar{\rho}$ columns report average normalized squared target distances, with the Base equal to one. Segment coefficients use true targets and are oracle diagnostics.

Dataset	H	N	p_+	p_{w}	p_{ws}	$\bar{\rho}_{\mathrm{end}}$	$\bar{\rho}_{\mathrm{seg}}$
DLinear state predictor							
ETTm1	96	11,425	37.15	97.30	34.42	5.032	0.940
	336	11,185	35.04	100.00	35.04	3.424	0.972
ETTm2	96	11,425	22.92	100.00	22.91	23.166	0.982
	336	11,185	9.33	100.00	9.33	14.104	0.994
ETTh1	96	2,785	26.10	100.00	26.10	6.602	0.987
	336	2,545	6.01	100.00	5.97	5.718	0.997
ETTh2	96	2,785	12.53	100.00	12.53	22.389	0.995
	336	2,545	2.59	100.00	2.59	11.008	1.000
Weather	96	10,444	61.32	97.33	58.62	3.663	0.954
	336	10,204	84.56	81.14	65.71	1.708	0.928
Electricity	96	5,165	100.00	100.00	100.00	2.022	0.970
	336	4,925	100.00	100.00	100.00	1.729	0.967
Traffic	96	3,413	100.00	97.28	97.28	1.454	0.964
	336	3,173	100.00	100.00	100.00	1.425	0.967
Solar	96	10,417	99.85	26.83	26.68	1.045	0.711
	336	10,177	100.00	11.48	11.48	0.925	0.734
iTransformer state predictor							
ETTm1	96	11,425	58.61	98.03	56.63	3.923	0.937
	336	11,185	56.38	99.79	56.13	2.845	0.960
ETTm2	96	11,425	46.76	100.00	46.71	29.115	0.954
	336	11,185	41.66	100.00	41.64	20.248	0.955
ETTh1	96	2,785	56.48	100.00	56.45	3.349	0.949
	336	2,545	52.65	100.00	52.61	2.754	0.973
ETTh2	96	2,785	48.01	100.00	47.94	20.075	0.951
	336	2,545	49.90	100.00	49.90	15.088	0.938
Weather	96	10,444	87.98	98.46	86.41	4.687	0.918
	336	10,204	91.10	95.02	86.10	3.396	0.921
Electricity	96	5,165	100.00	99.90	99.90	2.370	0.962
	336	4,925	100.00	100.00	100.00	1.846	0.944
Traffic	96	3,413	83.18	95.72	78.76	1.977	0.963
	336	3,173	97.23	100.00	97.23	1.847	0.972
Solar	96	10,417	97.30	57.52	54.80	1.669	0.752
	336	10,177	100.00	32.26	32.26	1.050	0.717

Table 6 reports dataset-averaged p_{w} values of 47.86%, 24.18%, and 36.14% for the default MLP, DLinear, and iTransformer, respectively, with corresponding p_{ws} values of 39.18%, 19.85%, and 32.11%. Thus, even after joint FUSE training, predicted future states can remain worse endpoints than encoded base forecasts. The lower rates relative to the independent diagnostic predictors are consistent with the additional forecasting supervision received during joint FUSE training. Moreover, in a substantial fraction of windows, the same displacement still contains a better intermediate state. These results extend Predicted State Deviation and the phenomenon of worse predicted endpoints with useful displacement directions beyond the independently trained diagnostic predictors.

B Analysis of the Training Objective

FUSE uses state supervision to learn future guidance and a forecasting loss to optimize the resulting forecast. We examine how these objectives relate to the guidance displacement, the decoded output, and the trainable parameters. For a training pair (X, Y), let $Y_0, Y \in \mathbb{R}^{H\times C}$ and $S_0, \widehat{S}, S^\star \in \mathbb{R}^{H\times d}$.

Table 6: State-geometry diagnostics of trained FUSE predictors, grouped by forecasting backbone. Both metrics are checkpoint-level percentages over test windows: p_{w} counts predicted endpoints farther from the target than Base, while p_{ws} additionally requires a better oracle point on the base-to-predicted segment. Values are averaged equally across four horizons and random seeds; Mean gives equal weight to eight datasets. Oracle segment coefficients use true targets and are not the learned shared α.

Dataset	Default MLP		DLinear		iTransformer	
	p_{w}	p_{ws}	p_{w}	p_{ws}	p_{w}	p_{ws}
ETTm1	47.52	30.45	28.90	22.03	33.69	25.89
ETTm2	62.50	41.87	30.07	19.65	31.40	25.21
ETTh1	33.90	31.49	27.85	24.08	27.20	24.59
ETTh2	62.41	50.60	16.53	13.16	34.85	30.33
Weather	45.65	32.01	26.19	17.04	40.51	33.82
Electricity	27.26	27.26	3.53	3.53	26.31	26.31
Traffic	81.48	79.48	45.35	45.25	76.60	73.09
Solar	22.17	20.25	15.03	14.02	18.56	17.62
Mean	47.86	39.18	24.18	19.85	36.14	32.11

The relevant quantities are

$$\begin{aligned} Y_0 &= F(X), & S_0 &= E(Y_0), \\ S^\star &= E(Y), & \widehat{S} &= P_\theta(E(X)), \\ \Delta S &= \widehat{S} - S_0, & S_{\mathrm{ref}} &= S_0 + \alpha \Delta S, \\ R_0 &= Y_0 - D(S_0), & Y_{\mathrm{ref}} &= D(S_{\mathrm{ref}}) + (1+\beta) R_0. \end{aligned} \tag{36}$$

The base forecaster F, encoder E, and decoder D are fixed throughout FUSE training. The trainable quantities are the predictor parameters θ and the independent scalar coefficients $\alpha, \beta \in \mathbb{R}$. Each coefficient is shared across the inputs to a trained FUSE model. The following losses are written for one training pair and are averaged over the batch during training.

B.1 State Supervision for Future Guidance

The State MSE loss supervises the predicted state against the encoded future:

$$\mathcal{L}_{\mathrm{mse}} = \frac{1}{Hd} \|\widehat{S} - S^\star\|_F^2. \tag{37}$$

Although refinement uses a displacement from S_0, this loss also directly measures the error in that displacement. To see this, define the target displacement as $\Delta S^\star = S^\star - S_0$. Both displacements have the same starting state, so

$$\begin{aligned} \Delta S - \Delta S^\star &= (\widehat{S} - S_0) - (S^\star - S_0) \\ &= \widehat{S} - S^\star. \end{aligned} \tag{38}$$

Taking squared Frobenius norms gives

$$\frac{1}{Hd} \|\Delta S - \Delta S^\star\|_F^2 = \mathcal{L}_{\mathrm{mse}}. \tag{39}$$

Thus, State MSE supervision also constrains the discrepancy between the predicted and target displacements. An additional MSE loss on these displacements would duplicate the same objective.

The State cosine loss complements this supervision with an additional angular penalty. Let $\widehat{s}_t, s_t^\star \in \mathbb{R}^d$ be the predicted and target state vectors at future time t. We use

$$\begin{aligned} c_\epsilon(a,b) &= \frac{a^\top b}{\max\{\|a\|_2, \epsilon\} \max\{\|b\|_2, \epsilon\}}, \\ \mathcal{L}_{\cos} &= \frac{1}{H} \sum_{t=1}^{H} \left[1 - c_\epsilon(\widehat{s}_t, s_t^\star)\right], \qquad \epsilon = 10^{-8}. \end{aligned} \tag{40}$$

Thus, cosine similarity is computed across the state dimensions at each time step and then averaged over time. Together, the two terms define the State prediction loss:

$$\mathcal{L}_{\text{state}} = \lambda_{\text{m}}\mathcal{L}_{\text{mse}} + \lambda_{\text{c}}\mathcal{L}_{\cos}. \tag{41}$$

The role of the additional cosine term follows from the Euclidean error decomposition. For a time step with $r_t = \|\widehat{s}_t\|_2 \geq \epsilon$ and $r_t^\star = \|s_t^\star\|_2 \geq \epsilon$, let ϑ_t denote the angle between the two state vectors. Since $\widehat{s}_t^\top s_t^\star = r_t r_t^\star \cos\vartheta_t$,

$$\begin{aligned}\|\widehat{s}_t - s_t^\star\|_2^2 &= r_t^2 + (r_t^\star)^2 - 2r_t r_t^\star \cos\vartheta_t \\ &= (r_t - r_t^\star)^2 + 2r_t r_t^\star(1 - \cos\vartheta_t).\end{aligned} \tag{42}$$

When this norm condition holds at all time steps, substituting into Equation 41 yields

$$\mathcal{L}_{\text{state}} = \frac{1}{H}\sum_{t=1}^{H}\left[\frac{\lambda_{\text{m}}}{d}(r_t - r_t^\star)^2 + \left(\frac{2\lambda_{\text{m}}}{d}r_t r_t^\star + \lambda_{\text{c}}\right)(1 - \cos\vartheta_t)\right]. \tag{43}$$

State MSE already penalizes both norm and angular discrepancies. The additional cosine term increases the angular penalty by λ_{c}, independently of the product of the state norms. For smaller norms, Equation 40 defines the stabilized loss directly. The angle above is between the absolute state vectors $\widehat{s}_t$ and $s_t^\star$. Cosine similarity is not invariant to subtracting the base state $s_{0,t}$, so this term does not directly supervise the angle between the two displacements.

We next examine how the state prediction error measured by State MSE affects the decoded output. We first establish a bound for the frozen decoder. Our decoder applies the same two linear layers and a ReLU activation to each time step:

$$D(z) = W_2\sigma(W_1 z + b_1) + b_2, \qquad \sigma(u) = \max\{u, 0\}, \tag{44}$$

where σ acts elementwise. The spectral norms of the weight matrices give a Lipschitz upper bound for such networks (Virmaux & Scaman, 2018, Proposition 1). For completeness, the bound follows directly from $|\sigma(a) - \sigma(b)| \leq |a - b|$ for scalar inputs. Applying this inequality to each coordinate yields $\|\sigma(u) - \sigma(v)\|_2 \leq \|u - v\|_2$. Consequently, for any two state vectors $z, z' \in \mathbb{R}^d$,

$$\begin{aligned}\|D(z) - D(z')\|_2 &= \big\|W_2\big[\sigma(W_1 z + b_1) - \sigma(W_1 z' + b_1)\big]\big\|_2 \\ &\leq \|W_2\|_2\,\|\sigma(W_1 z + b_1) - \sigma(W_1 z' + b_1)\|_2 \\ &\leq \|W_2\|_2\|W_1(z - z')\|_2 \\ &\leq \|W_2\|_2\|W_1\|_2\|z - z'\|_2.\end{aligned} \tag{45}$$

Here, $\|\cdot\|_2$ denotes the spectral norm when applied to a matrix. Let $K_D = \|W_2\|_2\|W_1\|_2$. For state sequences $U, V \in \mathbb{R}^{H\times d}$, denote their state vectors at time t by u_t and v_t. Summing the squared vector bounds over the horizon gives

$$\begin{aligned}\|D(U) - D(V)\|_F^2 &= \sum_{t=1}^{H}\|D(u_t) - D(v_t)\|_2^2 \\ &\leq K_D^2\sum_{t=1}^{H}\|u_t - v_t\|_2^2 \\ &= K_D^2\|U - V\|_F^2.\end{aligned} \tag{46}$$

With this decoder bound, we can isolate the effect of state prediction error on the refined output. For fixed Y_0, E, D, α, and β, define

$$\mathcal{Q}_{\alpha,\beta}(S) = D\big(S_0 + \alpha(S - S_0)\big) + (1 + \beta)R_0. \tag{47}$$

The actual output is $\mathcal{Q}_{\alpha,\beta}(\widehat{S}) = Y_{\text{ref}}$. Replacing $\widehat{S}$ by $S^\star$ defines a mathematical reference output with the same coefficients and residual. The difference between their decoder inputs is

$$\big[S_0 + \alpha(\widehat{S} - S_0)\big] - \big[S_0 + \alpha(S^\star - S_0)\big] = \alpha(\widehat{S} - S^\star). \tag{48}$$

The residual terms cancel when subtracting the outputs. Applying Equation 46 therefore gives

$$\begin{aligned}\|\mathcal{Q}_{\alpha,\beta}(\widehat{S}) - \mathcal{Q}_{\alpha,\beta}(S^\star)\|_F &\le K_D\|\alpha(\widehat{S} - S^\star)\|_F \\ &= K_D|\alpha|\|\widehat{S} - S^\star\|_F.\end{aligned} \tag{49}$$

After squaring and dividing by HC, the bound becomes

$$\begin{aligned}\frac{1}{HC}\|\mathcal{Q}_{\alpha,\beta}(\widehat{S}) - \mathcal{Q}_{\alpha,\beta}(S^\star)\|_F^2 &\le \frac{K_D^2\alpha^2}{HC}\|\widehat{S} - S^\star\|_F^2 \\ &= K_D^2\alpha^2\frac{d}{C}\,\mathcal{L}_{\text{mse}}.\end{aligned} \tag{50}$$

This relation bounds the output discrepancy introduced by using a predicted state in place of the target state. It holds for any real α, including reverse movement and extrapolation. Its reference is $\mathcal{Q}_{\alpha,\beta}(S^\star)$, which need not equal Y.

B.2 Forecasting Loss for Refinement

State supervision constrains the guidance, whereas the forecasting loss evaluates the complete output produced from that guidance. Define the changes contributed by state refinement and residual adjustment as

$$A = D(S_{\text{ref}}) - D(S_0), \qquad B = \beta R_0. \tag{51}$$

Since $D(S_0) + R_0 = Y_0$, the output can be rewritten as

$$\begin{aligned}Y_{\text{ref}} &= D(S_{\text{ref}}) + (1+\beta)R_0 \\ &= D(S_0) + R_0 + \big[D(S_{\text{ref}}) - D(S_0)\big] + \beta R_0 \\ &= Y_0 + A + B.\end{aligned} \tag{52}$$

Writing the base forecasting error as $e_0 = Y - Y_0$, we obtain

$$\begin{aligned}\mathcal{L}_{\text{forecast}} &= \frac{1}{HC}\|Y_{\text{ref}} - Y\|_F^2 \\ &= \frac{1}{HC}\|Y_0 + A + B - Y\|_F^2 \\ &= \frac{1}{HC}\|e_0 - (A+B)\|_F^2.\end{aligned} \tag{53}$$

The forecasting loss therefore trains the combined output change $A + B$ to compensate for the error of the base forecast. It evaluates the decoded state change and the adjusted residual together.

The need for this loss remains even when state prediction is exact. To establish this, define the reconstruction residual of the target as

$$R^\star = Y - D(S^\star), \qquad Y = D(S^\star) + R^\star. \tag{54}$$

Substituting this expression for Y gives

$$\begin{aligned}Y - Y_{\text{ref}} &= D(S^\star) + R^\star - D(S_{\text{ref}}) - (1+\beta)R_0 \\ &= \big[D(S^\star) - D(S_{\text{ref}})\big] + \big[R^\star - (1+\beta)R_0\big].\end{aligned} \tag{55}$$

If $\widehat{S} = S^\star$ and $\alpha = 1$, then $S_{\text{ref}} = S_0 + (S^\star - S_0) = S^\star$. The first term vanishes, leaving

$$Y - Y_{\text{ref}} = R^\star - (1+\beta)R_0. \tag{56}$$

Hence, reaching the exact target state leaves a forecasting error whenever the retained residual differs from the target residual. A single scalar β can remove this discrepancy only when $R^\star$ is the corresponding scalar multiple of R_0. State prediction loss and forecasting loss consequently have distinct roles: the former supervises the predicted future state, while the latter optimizes how state refinement and residual adjustment jointly reconstruct the future observations.

B.3 JOINT OPTIMIZATION

FUSE optimizes the future state predictor together with the state refinement coefficient α and residual adjustment coefficient β. The joint loss for a training pair is

$$\mathcal{L} = \mathcal{L}_{\text{forecast}} + \underbrace{\lambda_{\text{m}}\mathcal{L}_{\text{mse}} + \lambda_{\text{c}}\mathcal{L}_{\cos}}_{\mathcal{L}_{\text{state}}}. \tag{57}$$

The loss weights $\lambda_{\text{m}}, \lambda_{\text{c}} \geq 0$ remain fixed during training. We optimize θ, α, β while keeping F, E, and D frozen.

The gradient paths explain how the joint loss trains the predictor and the two coefficients. Consider a differentiable forward pass for one training pair, with any stochastic masks held fixed. Let $\text{vec}(\cdot)$ stack a matrix into a vector using a fixed ordering, and let p be the number of predictor parameters. Define

$$\begin{aligned} J_P &= \frac{\partial \,\text{vec}(\widehat{S})}{\partial \theta} \in \mathbb{R}^{Hd \times p}, \\ J_D &= \left.\frac{\partial \,\text{vec}(D(S))}{\partial \,\text{vec}(S)}\right|_{S=S_{\text{ref}}} \in \mathbb{R}^{HC \times Hd}, \\ g &= \frac{\partial \mathcal{L}_{\text{forecast}}}{\partial \,\text{vec}(Y_{\text{ref}})} = \frac{2}{HC}\,\text{vec}(Y_{\text{ref}} - Y). \end{aligned} \tag{58}$$

Because S_0 and R_0 are fixed with respect to θ, α, β, differentiating the refinement equation gives

$$\begin{aligned} \frac{\partial \,\text{vec}(S_{\text{ref}})}{\partial \theta} &= \alpha J_P, \\ \frac{\partial \,\text{vec}(S_{\text{ref}})}{\partial \alpha} &= \text{vec}(\Delta S), \\ \frac{\partial \,\text{vec}(S_{\text{ref}})}{\partial \beta} &= 0. \end{aligned} \tag{59}$$

Passing these derivatives through the decoder and differentiating the residual term yields

$$\begin{aligned} \frac{\partial \,\text{vec}(Y_{\text{ref}})}{\partial \theta} &= \alpha J_D J_P, \\ \frac{\partial \,\text{vec}(Y_{\text{ref}})}{\partial \alpha} &= J_D\,\text{vec}(\Delta S), \\ \frac{\partial \,\text{vec}(Y_{\text{ref}})}{\partial \beta} &= \text{vec}(R_0). \end{aligned} \tag{60}$$

Applying the chain rule to the scalar forecasting loss then gives

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\text{forecast}} &= \alpha J_P^\top J_D^\top g, \\ \frac{\partial \mathcal{L}_{\text{forecast}}}{\partial \alpha} &= g^\top J_D\,\text{vec}(\Delta S), \\ \frac{\partial \mathcal{L}_{\text{forecast}}}{\partial \beta} &= g^\top \text{vec}(R_0). \end{aligned} \tag{61}$$

Freezing the decoder parameters leaves J_D available for differentiation with respect to its input. Thus, forecasting supervision reaches the future state predictor through the frozen decoder whenever this path has a nonzero gradient.

The state losses depend directly on $\widehat{S}$ and are independent of α and β. Their gradients add to the predictor update:

$$\begin{aligned} \nabla_\theta \mathcal{L} &= \alpha J_P^\top J_D^\top g + \lambda_{\text{m}} \nabla_\theta \mathcal{L}_{\text{mse}} + \lambda_{\text{c}} \nabla_\theta \mathcal{L}_{\cos}, \\ \frac{\partial \mathcal{L}}{\partial \alpha} &= \frac{\partial \mathcal{L}_{\text{forecast}}}{\partial \alpha}, \\ \frac{\partial \mathcal{L}}{\partial \beta} &= \frac{\partial \mathcal{L}_{\text{forecast}}}{\partial \beta}. \end{aligned} \tag{62}$$

In particular, differentiating State MSE gives

$$\nabla_\theta \mathcal{L}_{\text{mse}} = \frac{2}{Hd} J_P^\top \operatorname{vec}(\widehat{S} - S^\star). \tag{63}$$

This direct supervision path contains no factor of α.

FUSE initializes both coefficients to zero. Substituting $\alpha = \beta = 0$ into the forward equations gives

$$S_{\text{ref}} = S_0, \qquad Y_{\text{ref}} = D(S_0) + R_0 = Y_0. \tag{64}$$

At initialization, the output therefore retains the base forecast exactly. Equation 61 also gives $\nabla_\theta \mathcal{L}_{\text{forecast}} = 0$ when $\alpha = 0$. The joint predictor gradient at this point is

$$\nabla_\theta \mathcal{L}|_{\alpha=\beta=0} = \lambda_{\text{m}} \nabla_\theta \mathcal{L}_{\text{mse}} + \lambda_{\text{c}} \nabla_\theta \mathcal{L}_{\cos}. \tag{65}$$

State supervision therefore provides a direct training signal to the predictor from initialization. The forecasting gradient for α has no multiplicative factor of α, so it may already be nonzero at initialization. As α changes, the forecasting loss can also update the predictor through state refinement. At ReLU kinks, automatic differentiation uses the implementation's chosen derivative. The identity output and the independence of that output from θ at $\alpha = 0$ hold regardless of this choice. Together, these relations explain how the joint objective learns future guidance while optimizing its use in the final forecast.

To interpret the fixed loss weights, consider forecasting risk minimization with constraints on state prediction error. Let the training set contain n pairs and define the empirical risks

$$\mathcal{R}_j = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_j(X_i, Y_i), \qquad j \in \{\text{forecast}, \text{mse}, \cos\}. \tag{66}$$

The forecasting risk depends on θ, α, β, whereas the two state risks depend on θ. For prescribed tolerances $\tau_{\text{m}}, \tau_{\text{c}} \geq 0$, consider

$$\begin{aligned} \min_{\theta,\alpha,\beta} \quad & \mathcal{R}_{\text{forecast}}(\theta, \alpha, \beta) \\ \text{subject to} \quad & \mathcal{R}_{\text{mse}}(\theta) \leq \tau_{\text{m}}, \\ & \mathcal{R}_{\cos}(\theta) \leq \tau_{\text{c}}. \end{aligned} \tag{67}$$

The Lagrangian augments the objective with weighted constraint functions (Boyd & Vandenberghe, 2004, Section 5.1.1). For fixed multipliers $\lambda_{\text{m}}, \lambda_{\text{c}} \geq 0$, it is

$$\begin{aligned} \mathcal{J}(\theta, \alpha, \beta) &= \mathcal{R}_{\text{forecast}} + \lambda_{\text{m}}(\mathcal{R}_{\text{mse}} - \tau_{\text{m}}) + \lambda_{\text{c}}(\mathcal{R}_{\cos} - \tau_{\text{c}}) \\ &= \mathcal{R}_{\text{forecast}} + \lambda_{\text{m}} \mathcal{R}_{\text{mse}} + \lambda_{\text{c}} \mathcal{R}_{\cos} - \lambda_{\text{m}} \tau_{\text{m}} - \lambda_{\text{c}} \tau_{\text{c}}. \end{aligned} \tag{68}$$

The last two terms are constant with respect to all trainable parameters. Removing them preserves the minimizers of $\mathcal{J}$ for these fixed multipliers. The remaining terms equal the empirical average of the joint loss in Equation 57. This provides a soft constraint interpretation of the state supervision terms. FUSE uses fixed weights rather than optimizing dual variables. Their values control the tradeoff between the objectives and are selected empirically. The construction does not imply that arbitrary fixed weights enforce the prescribed tolerances or solve the constrained problem.

State supervision constrains future guidance, while forecasting supervision optimizes its use together with reconstruction residual adjustment. Supervising the predicted state does not require the refined state to coincide with it, allowing interpolation, extrapolation, and reversal.

C Experimental Details

C.1 Dataset Descriptions

We evaluate FUSE on eight multivariate forecasting benchmarks from five dataset families. Table 7 summarizes their dimensions, sampling intervals, and splits.

- **ETT** (Zhou et al., 2021) records transformer oil temperature and six power load variables at two locations. ETTh1 and ETTh2 contain hourly observations, while ETTm1 and ETTm2 are sampled every 15 minutes.

- **Weather** (Wu et al., 2023) contains 21 meteorological variables recorded every 10 minutes in Germany, including temperature, pressure, and humidity.
- **Electricity** (Wu et al., 2023) contains the hourly electricity consumption of 321 clients.
- **Traffic** (Wu et al., 2023) records hourly road occupancy rates from 862 sensors in the San Francisco Bay Area.
- **Solar** (Liu et al., 2024) contains solar power generation measurements from 137 photovoltaic plants in Alabama, sampled every 10 minutes.

Table 7: **Dataset statistics.** "Dim" denotes the number of variables. "Dataset Size" reports the numbers of time points in the (training, validation, test) splits. Prediction lengths are measured in time steps. For ETT, the counts cover the interval used in our experiments.

Dataset	Dim	Prediction Length	Dataset Size	Frequency
ETTh1	7	$\{96, 192, 336, 720\}$	$(8640, 2880, 2880)$	1 hour
ETTh2	7	$\{96, 192, 336, 720\}$	$(8640, 2880, 2880)$	1 hour
ETTm1	7	$\{96, 192, 336, 720\}$	$(34560, 11520, 11520)$	15 min
ETTm2	7	$\{96, 192, 336, 720\}$	$(34560, 11520, 11520)$	15 min
Weather	21	$\{96, 192, 336, 720\}$	$(36887, 5270, 10539)$	10 min
Electricity	321	$\{96, 192, 336, 720\}$	$(18412, 2632, 5260)$	1 hour
Traffic	862	$\{96, 192, 336, 720\}$	$(12280, 1756, 3508)$	1 hour
Solar	137	$\{96, 192, 336, 720\}$	$(36792, 5256, 10512)$	10 min

C.2 Baselines and Evaluation Protocol

We compare FUSE with nine forecasting methods, described below.

- **TimeAlign** (Hu et al., 2026): TimeAlign uses an auxiliary reconstruction task to learn representations of future observations. An alignment objective brings the prediction representations toward these reconstruction representations to address distribution shifts between history and future. Official code: `https://github.com/TROUBADOUR000/TimeAlign`.
- **PDF** (Dai et al., 2024): PDF reorganizes time series into two-dimensional representations according to multiple periods. It models short-term changes and long-term dependencies with separate components, then aggregates the resulting features for forecasting. Official code: `https://github.com/Hank0626/PDF`.
- **CMoS** (Si et al., 2025): CMoS directly models spatial correlations between time series chunks. Its correlation mixing mechanism combines different correlation structures, while an optional periodicity injection component incorporates recurring temporal patterns. Official code: `https://github.com/CSTCloudOps/CMoS`.
- **TimeBase** (Huang et al., 2025): TimeBase represents time series through a small set of temporal basis components. By exploiting the low-rank structure of the series, it replaces prediction over individual time points with prediction over segments. Official code: `https://github.com/hqh0728/TimeBase`.
- **iTransformer** (Liu et al., 2024): iTransformer embeds the history of each variable as a token. Attention captures dependencies across variables, while feed-forward networks transform each variable token to learn temporal representations for forecasting. Official code: `https://github.com/thuml/iTransformer`.
- **PatchTST** (Nie et al., 2023): PatchTST groups consecutive time points into patches and processes each variable independently with a shared Transformer. Patching retains local temporal information and reduces the number of tokens used to model longer histories. Official code: `https://github.com/yuqinie98/PatchTST`.

- **TimeMixer** (Wang et al., 2024a): TimeMixer constructs representations at multiple temporal resolutions and decomposes them into seasonal and trend components. MLP blocks mix seasonal information from fine to coarse scales and trend information in the opposite direction. Forecasts from multiple scales are then combined. Official code: `https://github.com/kwuking/TimeMixer`.
- **TimesNet** (Wu et al., 2023): TimesNet identifies dominant periods and reshapes a time series into multiple two-dimensional tensors. Inception-style convolutional blocks capture variations within and across periods, and the resulting representations are aggregated for prediction. Official code: `https://github.com/thuml/TimesNet`.
- **DLinear** (Zeng et al., 2023): DLinear separates the input into a smooth trend and a remainder using moving averages. Two linear mappings independently project these components onto the forecasting horizon, and their outputs are summed to obtain the prediction. Official code: `https://github.com/cure-lab/LTSF-Linear`.

For the plug-in experiments, we evaluate FUSE with the default MLP, DLinear, and iTransformer. Each base forecaster is first trained independently and then kept frozen during FUSE training. We compare its original forecast with the refined forecast produced by FUSE. For DLinear and iTransformer, our experiments use the implementations in the Time Series Library (Wu et al., 2023), available at `https://github.com/thuml/Time-Series-Library`.

We consider forecast horizons $H \in \{96, 192, 336, 720\}$ and lookback lengths $L \in \{336, 512, 720\}$. Forecasting accuracy is measured by mean squared error (MSE) and mean absolute error (MAE). For N_{test} test windows, the metrics are

$$\begin{aligned} \text{MSE} &= \frac{1}{N_{\text{test}} HC} \sum_{i=1}^{N_{\text{test}}} \left\| \widehat{Y}^{(i)} - Y^{(i)} \right\|_F^2, \\ \text{MAE} &= \frac{1}{N_{\text{test}} HC} \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{H} \sum_{c=1}^{C} \left| \widehat{Y}^{(i)}_{t,c} - Y^{(i)}_{t,c} \right|, \end{aligned} \tag{69}$$

where $\widehat{Y}^{(i)}$ and $Y^{(i)}$ denote the forecast and target for window i.

Within each forecasting training run, checkpoints are selected by validation MSE. For the performance tables, we search the lookback length over $L \in \{336, 512, 720\}$ for each method, dataset, and forecasting horizon, and report the best-performing configuration. MAE is reported from the same selected run, and lookback selection is performed independently for each method. Rows marked Avg. report arithmetic means over the four forecast horizons. The plug-in table reports the mean relative reduction in the two horizon-averaged metrics:

$$\text{Gain} = \frac{1}{2} \left[\frac{\overline{\text{MSE}}_{\text{base}} - \overline{\text{MSE}}_{\text{FUSE}}}{\overline{\text{MSE}}_{\text{base}}} + \frac{\overline{\text{MAE}}_{\text{base}} - \overline{\text{MAE}}_{\text{FUSE}}}{\overline{\text{MAE}}_{\text{base}}} \right] \times 100\%. \tag{70}$$

Here, the overbar denotes averaging over horizons, and positive Gain indicates an improvement over the base forecaster.

C.3 Implementation Details

All experiments are implemented in PyTorch (Paszke et al., 2019) and run on a single NVIDIA RTX 4090 GPU with 24 GB memory. Unless otherwise specified, all experiments involving FUSE are repeated with three random seeds, and the reported FUSE results are averaged across the three runs. We pretrain one pointwise MLP autoencoder per dataset using Adam (Kingma & Ba, 2015). The base forecaster and FUSE are trained in separate stages with AdamW (Loshchilov & Hutter, 2019). During FUSE training, the base forecaster and autoencoder remain frozen. Base forecasting uses cosine learning rate decay (Loshchilov & Hutter, 2017). FUSE uses a constant learning rate with the default MLP and cosine decay with DLinear and iTransformer. Checkpoints are selected by validation reconstruction MAE for the autoencoder and validation forecasting MSE for forecasting models. Tables 8 and 9 summarize the key training settings and dataset configurations for the default model.

Table 8: **Common hyperparameters.** Epoch budgets are maximum values. Predictor width denotes both the embedding and feed-forward dimensions: 64 for $H = 96$ and 160 for the other horizons.

Parameter	Value	Parameter	Value
Training and Objectives			
AE epochs / patience	$500/20$	Base epochs / patience	$50/8$
FUSE epochs / patience	$120/12$	FUSE learning rate	5×10^{-4}
Loss weights $(\lambda_{\mathrm{m}}, \lambda_{\mathrm{c}})$	$(10, 15)$	Initial (α, β)	$(0, 0)$
Autoencoder and MLP Configuration			
AE window length	24	AE hidden dimension	$2d$
AE reconstruction loss	MAE	Default MLP blocks	2
Predictor MLP blocks	2	Predictor dropout	0.1

Table 9: **Dataset-specific hyperparameters for the default model.** Here, d is the AE state dimension, while d_f and d_{ff} are the embedding and feed-forward dimensions of the base MLP. Learning rates refer to base training, and batch pairs give (base, FUSE) training batch sizes. At $H = 720$, ETTm1 uses $d_f = 256$ and Weather uses $d_{\mathrm{ff}} = 128$; other values remain unchanged.

Dataset	d	d_f	d_{ff}	Learning rate	Batch size
ETTh1	32	32	32	5×10^{-4}	$(256, 512)$
ETTh2	64	32	32	5×10^{-4}	$(256, 512)$
ETTm1	32	128	256	10^{-4}	$(256, 512)$
ETTm2	64	128	128	10^{-4}	$(256, 512)$
Weather	64	128	256	10^{-4}	$(32, 32)$
Electricity	512	512	2048	5×10^{-4}	$(16, 16)$
Traffic	512	512	2048	10^{-3}	$(16, 16)$
Solar	256	256	256	5×10^{-4}	$(32, 32)$

D Full Results

D.1 Forecasting Performance

Table 10 provides the complete forecasting results corresponding to Table 1. For each method, dataset, and forecasting horizon, we select the lookback length from $L \in \{336, 512, 720\}$ with the lowest test MSE. MAE is reported from the same run. FUSE uses the default MLP as its base forecaster.

D.2 Full Plug-in Results

Table 11 reports the plug-in results for the default MLP, DLinear, and iTransformer at each forecasting horizon. For each backbone, we compare the base forecast with its FUSE refinement using MSE, MAE, and Gain. The Avg. rows report errors averaged over the four horizons, with Gain computed from these averages, matching Table 2. Across the 96 combinations of datasets, backbones, and horizons, FUSE reduces MSE in 94 and achieves positive Gain in 91.

D.3 Analysis of Forecast Refinement

This section defines the three components used to analyze forecast refinement and their contributions to MSE reduction. The analysis applies to the complete change from Y_0 to Y_{ref}. Since $Y_0 = D(S_0) + R_0$, the output equation gives

$$\begin{aligned} Y_{\mathrm{ref}} - Y_0 &= D(S_{\mathrm{ref}}) + (1 + \beta)R_0 - \big[D(S_0) + R_0\big] \\ &= D(S_{\mathrm{ref}}) - D(S_0) + \beta R_0. \end{aligned} \tag{71}$$

Thus, the analyzed change includes both decoded state refinement and reconstruction residual adjustment. The components below describe their combined effect in observation space.

Table 10: Full results of long-term forecasting. MSE and MAE are reported for each forecasting horizon, and Avg. denotes the average over $H \in \{96, 192, 336, 720\}$. The best and second-best results for each metric are marked in **bold** and underline, respectively, including ties.

Models		FUSE (Ours)		TimeAlign (2026)		CMoS (2025)		TimeBase (2025)		PDF (2024)		iTransformer (2024)		TimeMixer (2024a)		PatchTST (2023)		TimesNet (2023)		DLinear (2023)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.274**	**0.327**	0.279	0.330	0.292	0.345	0.311	0.351	0.286	0.340	0.300	0.353	0.293	0.345	0.293	0.346	0.338	0.375	0.300	0.345
	192	**0.315**	**0.354**	0.322	0.356	0.334	0.366	0.338	0.371	0.321	0.364	0.345	0.382	0.335	0.372	0.333	0.370	0.371	0.387	0.336	0.366
	336	**0.346**	0.374	0.350	0.375	0.366	0.386	0.364	0.386	0.354	0.383	0.374	0.398	0.368	0.386	0.369	**0.369**	0.410	0.411	0.367	0.386
	720	**0.402**	0.408	0.407	**0.406**	0.425	0.417	0.413	0.414	0.408	0.415	0.429	0.430	0.426	0.417	0.416	0.420	0.478	0.450	0.419	0.416
	Avg.	**0.334**	**0.366**	0.340	0.367	0.354	0.379	0.357	0.381	0.342	0.376	0.362	0.391	0.356	0.380	0.353	0.376	0.399	0.406	0.356	0.378
ETTm2	96	**0.156**	**0.242**	0.157	**0.242**	0.167	0.257	0.162	0.256	0.163	0.251	0.175	0.266	0.165	0.256	0.166	0.256	0.189	0.265	0.164	0.255
	192	**0.213**	**0.282**	0.215	0.283	0.228	0.299	0.218	0.293	0.219	0.290	0.242	0.312	0.225	0.298	0.223	0.296	0.254	0.310	0.224	0.304
	336	**0.264**	**0.319**	0.266	0.322	0.273	0.325	0.270	0.328	0.269	0.330	0.282	0.340	0.277	0.332	0.274	0.329	0.313	0.345	0.277	0.337
	720	0.348	**0.374**	**0.347**	0.376	0.367	0.385	0.352	0.380	0.349	0.382	0.378	0.398	0.360	0.387	0.362	0.385	0.413	0.402	0.371	0.401
	Avg.	**0.245**	**0.304**	0.246	0.306	0.259	0.317	0.251	0.314	0.250	0.313	0.269	0.329	0.257	0.318	0.256	0.317	0.292	0.331	0.259	0.324
ETTh1	96	0.359	0.384	0.362	0.387	0.361	**0.383**	**0.349**	0.384	0.360	0.391	0.386	0.405	0.372	0.401	0.377	0.397	0.384	0.402	0.379	0.403
	192	0.390	0.412	0.404	0.412	0.405	**0.409**	**0.387**	0.410	0.392	0.414	0.424	0.440	0.413	0.430	0.409	0.428	0.557	0.436	0.408	0.419
	336	0.410	0.421	0.424	0.430	0.412	0.420	**0.408**	**0.418**	0.418	0.435	0.449	0.460	0.438	0.450	0.431	0.444	0.491	0.469	0.440	0.440
	720	**0.432**	0.455	0.433	0.451	0.433	0.451	0.439	**0.446**	0.456	0.462	0.495	0.487	0.486	0.484	0.457	0.477	0.521	0.500	0.471	0.493
	Avg.	0.398	0.418	0.406	0.420	0.403	0.416	**0.396**	**0.415**	0.407	0.426	0.439	0.448	0.427	0.441	0.419	0.437	0.488	0.452	0.425	0.439
ETTh2	96	0.269	0.331	**0.266**	**0.328**	0.274	0.341	0.292	0.345	0.276	0.341	0.297	0.348	0.281	0.351	0.274	0.337	0.334	0.370	0.289	0.353
	192	**0.332**	0.376	0.335	**0.374**	0.333	0.383	0.339	0.387	0.339	0.382	0.371	0.403	0.349	0.387	0.348	0.384	0.404	0.413	0.383	0.418
	336	0.360	0.397	0.363	0.399	**0.342**	**0.384**	0.358	0.410	0.374	0.406	0.404	0.428	0.366	0.413	0.377	0.416	0.389	0.435	0.448	0.465
	720	0.387	**0.423**	0.388	0.428	**0.374**	**0.423**	0.400	0.448	0.398	0.433	0.424	0.444	0.401	0.436	0.406	0.441	0.434	0.448	0.605	0.551
	Avg.	0.337	**0.382**	0.338	**0.382**	**0.331**	0.383	0.347	0.398	0.347	0.391	0.374	0.406	0.349	0.397	0.351	0.395	0.390	0.417	0.431	0.447
Weather	96	**0.142**	0.189	0.144	**0.183**	0.144	0.193	0.146	0.198	0.147	0.196	0.159	0.208	0.147	0.198	0.149	0.198	0.172	0.220	0.170	0.230
	192	**0.184**	0.229	**0.184**	**0.220**	0.186	0.237	0.185	0.241	0.193	0.240	0.200	0.248	0.192	0.243	0.194	0.241	0.219	0.261	0.216	0.273
	336	**0.234**	0.269	0.238	**0.268**	0.240	0.281	0.236	0.281	0.245	0.280	0.253	0.289	0.247	0.284	0.245	0.282	0.280	0.306	0.258	0.307
	720	**0.308**	0.323	0.311	**0.322**	0.311	0.332	0.309	0.331	0.323	0.334	0.321	0.338	0.318	0.330	0.314	0.334	0.365	0.359	0.323	0.362
	Avg.	**0.217**	0.253	0.219	**0.248**	0.220	0.261	0.219	0.263	0.227	0.263	0.233	0.271	0.226	0.264	0.226	0.264	0.259	0.287	0.242	0.293
Electricity	96	**0.126**	**0.216**	**0.126**	**0.216**	0.129	0.223	0.139	0.231	0.128	0.222	0.138	0.237	0.142	0.247	0.129	0.222	0.168	0.272	0.140	0.237
	192	0.144	**0.234**	0.146	0.236	**0.142**	0.236	0.153	0.245	0.147	0.242	0.157	0.256	0.168	0.269	0.147	0.240	0.184	0.289	0.152	0.249
	336	**0.156**	**0.249**	0.158	**0.249**	0.161	0.254	0.169	0.262	0.165	0.260	0.167	0.264	0.171	0.260	0.163	0.259	0.198	0.300	0.170	0.267
	720	**0.182**	**0.276**	0.189	0.278	0.200	0.288	0.207	0.294	0.199	0.289	0.194	0.286	0.209	0.313	0.197	0.290	0.220	0.320	0.203	0.301
	Avg.	**0.152**	**0.244**	0.155	0.245	0.158	0.250	0.167	0.258	0.160	0.253	0.164	0.261	0.173	0.272	0.159	0.253	0.193	0.295	0.166	0.264
Traffic	96	**0.351**	**0.231**	0.354	0.233	0.367	0.256	0.394	0.267	0.368	0.252	0.363	0.265	0.369	0.257	0.370	0.262	0.593	0.321	0.395	0.275
	192	**0.378**	0.247	0.381	**0.242**	0.379	0.261	0.403	0.271	0.382	0.261	0.385	0.273	0.400	0.272	0.386	0.269	0.617	0.336	0.407	0.280
	336	0.392	0.268	**0.388**	**0.255**	0.397	0.270	0.417	0.278	0.393	0.268	0.396	0.277	0.407	0.272	0.396	0.275	0.629	0.336	0.417	0.286
	720	**0.434**	0.272	**0.434**	**0.267**	0.442	0.295	0.456	0.298	0.438	0.297	0.445	0.308	0.461	0.316	0.435	0.295	0.640	0.350	0.454	0.308
	Avg.	**0.389**	0.255	**0.389**	**0.249**	0.396	0.271	0.418	0.279	0.395	0.270	0.397	0.281	0.409	0.279	0.397	0.275	0.620	0.336	0.418	0.287
Solar	96	**0.161**	**0.208**	0.175	0.211	0.188	0.241	0.220	0.269	0.181	0.247	0.188	0.232	0.179	0.232	0.178	0.229	0.219	0.314	0.197	0.210
	192	**0.187**	**0.212**	0.192	0.215	0.207	0.256	0.232	0.266	0.200	0.259	0.193	0.248	0.201	0.259	0.189	0.246	0.231	0.322	0.218	0.222
	336	0.199	**0.222**	0.209	0.226	0.219	0.263	0.247	0.274	0.208	0.269	0.203	0.249	**0.190**	0.256	0.198	0.249	0.246	0.337	0.234	0.231
	720	**0.198**	**0.226**	0.203	0.229	0.226	0.269	0.247	0.273	0.212	0.275	0.223	0.261	0.203	0.261	0.209	0.256	0.280	0.363	0.243	0.241
	Avg.	**0.186**	**0.217**	0.195	0.220	0.210	0.257	0.237	0.271	0.200	0.263	0.202	0.248	0.193	0.252	0.194	0.245	0.244	0.334	0.223	0.226

D.3.1 Decomposition Relative to the Base Forecast

For one test window and variable, let $b, o, y \in \mathbb{R}^H$ denote the base forecast, refined forecast, and target, respectively. Define the output change and base forecasting error as

$$u = o - b, \qquad e = y - b. \tag{72}$$

For any $z \in \mathbb{R}^H$, write $\mu_z = H^{-1}\mathbf{1}^\top z$, where $\mathbf{1} \in \mathbb{R}^H$ is the all-ones vector. The centered base waveform is

$$q = b - \mu_b \mathbf{1}, \qquad \mathbf{1}^\top q = 0. \tag{73}$$

The constant direction and the centered base waveform define the mean and scaling components. For $q \neq 0$, their projection operators and the remaining shape projection are

$$\Pi_\mu = \frac{\mathbf{1}\mathbf{1}^\top}{H}, \qquad \Pi_{\mathrm{sc}} = \frac{qq^\top}{\|q\|_2^2}, \qquad \Pi_{\mathrm{sh}} = I_H - \Pi_\mu - \Pi_{\mathrm{sc}}, \tag{74}$$

where I_H is the identity matrix. These operators depend only on the base forecast. We hold them fixed when comparing the base forecast, refined forecast, and target.

Table 11: Full plug-in results for horizons $H \in \{96, 192, 336, 720\}$. **Bold** indicates improved MSE or MAE, and gray indicates degradation relative to Base. Gain (%) averages the relative MSE and MAE reductions; positive gains are shown in **green bold**. Avg. reports horizon-averaged errors and their corresponding Gain, while Mean Gain averages the eight dataset gains.

Dataset	H	Default MLP					DLinear					iTransformer				
		Base		+FUSE		Gain (%)	Base		+FUSE		Gain (%)	Base		+FUSE		Gain (%)
		MSE	MAE	MSE	MAE		MSE	MAE	MSE	MAE		MSE	MAE	MSE	MAE	
ETTm1	96	0.281	0.336	**0.274**	**0.327**	**+2.53**	0.300	0.345	**0.280**	**0.337**	**+4.40**	0.300	0.353	**0.282**	**0.337**	**+5.32**
	192	0.322	0.357	**0.315**	**0.354**	**+1.48**	0.336	0.366	**0.319**	**0.360**	**+3.33**	0.345	0.382	**0.322**	**0.363**	**+5.86**
	336	0.351	0.374	**0.346**	0.374	**+0.66**	0.367	0.386	**0.354**	**0.380**	**+2.54**	0.374	0.398	**0.353**	**0.383**	**+4.71**
	720	0.452	0.406	**0.402**	0.408	**+5.33**	0.419	0.416	**0.412**	**0.414**	**+0.98**	0.429	0.430	**0.410**	**0.416**	**+3.83**
	Avg.	0.351	0.368	**0.334**	**0.366**	**+2.78**	0.356	0.378	**0.341**	**0.373**	**+2.68**	0.362	0.391	**0.342**	**0.375**	**+4.87**
ETTm2	96	0.166	0.262	**0.156**	**0.242**	**+6.89**	0.164	0.255	**0.161**	**0.252**	**+1.53**	0.175	0.266	**0.161**	**0.250**	**+7.11**
	192	0.233	0.292	**0.213**	**0.282**	**+6.00**	0.224	0.304	**0.215**	**0.291**	**+4.16**	0.242	0.312	**0.221**	**0.292**	**+7.58**
	336	0.286	0.318	**0.264**	0.319	**+3.77**	0.277	0.337	**0.267**	**0.329**	**+3.02**	0.282	0.340	**0.271**	**0.328**	**+3.73**
	720	0.369	0.376	**0.348**	**0.374**	**+3.02**	0.371	0.401	**0.349**	**0.381**	**+5.43**	0.378	0.398	**0.356**	**0.383**	**+4.80**
	Avg.	0.264	0.312	**0.245**	**0.304**	**+4.71**	0.259	0.324	**0.248**	**0.313**	**+3.82**	0.269	0.329	**0.252**	**0.313**	**+5.58**
ETTh1	96	0.369	0.394	**0.359**	**0.384**	**+2.60**	0.379	0.403	**0.368**	**0.396**	**+2.30**	0.386	0.405	**0.369**	**0.402**	**+2.60**
	192	0.409	0.415	**0.390**	**0.412**	**+2.76**	0.408	0.419	**0.399**	**0.416**	**+1.50**	0.424	0.440	**0.409**	**0.432**	**+2.67**
	336	0.430	0.436	**0.410**	**0.421**	**+4.01**	0.440	0.440	**0.428**	**0.438**	**+1.67**	0.449	0.460	**0.436**	**0.450**	**+2.59**
	720	0.453	0.460	**0.432**	**0.455**	**+2.83**	0.471	0.493	0.473	0.494	-0.28	0.495	0.487	**0.472**	0.488	**+2.25**
	Avg.	0.415	0.426	**0.398**	**0.418**	**+3.07**	0.425	0.439	**0.417**	**0.436**	**+1.23**	0.439	0.448	**0.422**	**0.443**	**+2.52**
ETTh2	96	0.271	0.327	**0.269**	0.331	-0.33	0.289	0.353	**0.269**	**0.330**	**+6.70**	0.297	0.348	**0.275**	**0.335**	**+5.53**
	192	0.334	0.375	**0.332**	0.376	**+0.13**	0.383	0.418	**0.354**	**0.395**	**+6.58**	0.371	0.403	**0.351**	**0.387**	**+4.76**
	336	0.358	0.401	0.360	**0.397**	**+0.33**	0.448	0.465	**0.385**	**0.421**	**+11.79**	0.404	0.428	**0.384**	**0.411**	**+4.55**
	720	0.405	0.433	**0.387**	**0.423**	**+3.43**	0.605	0.551	**0.469**	**0.485**	**+17.18**	0.424	0.444	**0.407**	**0.438**	**+2.70**
	Avg.	0.342	0.384	**0.337**	**0.382**	**+1.04**	0.431	0.447	**0.369**	**0.408**	**+11.55**	0.374	0.406	**0.354**	**0.392**	**+4.28**
Weather	96	0.162	0.198	**0.142**	**0.189**	**+8.63**	0.170	0.230	**0.150**	**0.203**	**+11.90**	0.159	0.208	**0.151**	**0.201**	**+4.35**
	192	0.196	0.225	**0.184**	0.229	**+2.05**	0.216	0.273	**0.199**	**0.246**	**+8.99**	0.200	0.248	**0.195**	**0.242**	**+2.31**
	336	0.246	0.269	**0.234**	0.269	**+2.30**	0.258	0.307	**0.242**	**0.280**	**+7.37**	0.253	0.289	**0.243**	**0.280**	**+3.50**
	720	0.320	0.320	**0.308**	0.323	**+1.35**	0.323	0.362	**0.310**	**0.333**	**+6.05**	0.321	0.338	**0.311**	**0.330**	**+2.67**
	Avg.	0.231	0.253	**0.217**	**0.253**	**+3.08**	0.242	0.293	**0.225**	**0.265**	**+8.13**	0.233	0.271	**0.225**	**0.264**	**+3.11**
Electricity	96	0.139	0.218	**0.126**	**0.216**	**+5.22**	0.140	0.237	**0.132**	**0.231**	**+4.05**	0.138	0.237	**0.129**	**0.224**	**+6.07**
	192	0.159	0.246	**0.144**	**0.234**	**+7.11**	0.152	0.249	**0.146**	**0.244**	**+3.02**	0.157	0.256	**0.147**	**0.243**	**+5.83**
	336	0.168	0.252	**0.156**	**0.249**	**+4.36**	0.170	0.267	**0.161**	**0.262**	**+3.49**	0.167	0.264	**0.159**	**0.259**	**+3.38**
	720	0.192	0.281	**0.182**	**0.276**	**+3.54**	0.203	0.301	**0.190**	**0.291**	**+4.74**	0.194	0.286	**0.178**	**0.275**	**+6.14**
	Avg.	0.165	0.249	**0.152**	**0.244**	**+4.98**	0.166	0.264	**0.157**	**0.257**	**+3.87**	0.164	0.261	**0.153**	**0.250**	**+5.37**
Traffic	96	0.364	0.245	**0.351**	**0.231**	**+4.71**	0.395	0.275	**0.391**	**0.272**	**+1.05**	0.363	0.265	**0.362**	0.266	-0.10
	192	0.381	0.255	**0.378**	**0.247**	**+1.99**	0.407	0.280	**0.403**	**0.271**	**+2.01**	0.385	0.273	**0.380**	0.277	-0.02
	336	0.409	0.261	**0.392**	0.268	**+0.74**	0.417	0.286	**0.415**	**0.274**	**+2.24**	0.396	0.277	**0.393**	0.279	-0.03
	720	0.451	0.287	**0.434**	**0.272**	**+4.60**	0.454	0.308	**0.451**	**0.306**	**+0.71**	0.445	0.308	**0.439**	**0.287**	**+4.02**
	Avg.	0.401	0.262	**0.389**	**0.255**	**+3.04**	0.418	0.287	**0.415**	**0.281**	**+1.49**	0.397	0.281	**0.394**	**0.277**	**+1.07**
Solar	96	0.187	0.212	**0.161**	**0.208**	**+7.98**	0.197	0.210	**0.158**	**0.209**	**+10.13**	0.188	0.232	**0.163**	**0.212**	**+10.95**
	192	0.209	0.221	**0.187**	**0.212**	**+7.20**	0.218	0.222	**0.196**	0.237	**+1.58**	0.193	0.248	**0.191**	**0.233**	**+3.38**
	336	0.217	0.224	**0.199**	**0.222**	**+4.48**	0.234	0.231	**0.210**	0.248	**+1.31**	0.203	0.249	**0.199**	**0.240**	**+2.71**
	720	0.224	0.229	**0.198**	**0.226**	**+6.39**	0.243	0.241	**0.207**	0.246	**+6.42**	0.223	0.261	**0.201**	**0.244**	**+8.17**
	Avg.	0.209	0.221	**0.186**	**0.217**	**+6.46**	0.223	0.226	**0.193**	0.235	**+4.75**	0.202	0.248	**0.189**	**0.232**	**+6.30**
Mean Gain						**+3.64**					**+4.69**					**+4.14**

The projections are mutually orthogonal. To see this, first note that Π_μ and Π_{sc} are symmetric, and

$$\Pi_\mu^2 = \frac{\mathbf{1}(\mathbf{1}^\top \mathbf{1})\mathbf{1}^\top}{H^2} = \Pi_\mu,$$

$$\Pi_{\mathrm{sc}}^2 = \frac{q(q^\top q)q^\top}{(q^\top q)^2} = \Pi_{\mathrm{sc}}.$$

Because $\mathbf{1}^\top q = 0$,

$$\Pi_\mu \Pi_{\mathrm{sc}} = \frac{\mathbf{1}(\mathbf{1}^\top q) q^\top}{H\|q\|_2^2} = 0, \qquad \Pi_{\mathrm{sc}} \Pi_\mu = 0.$$

It follows that

$$\begin{aligned} \Pi_{\mathrm{sh}}^2 &= (I_H - \Pi_\mu - \Pi_{\mathrm{sc}})^2 \\ &= I_H - \Pi_\mu - \Pi_{\mathrm{sc}} = \Pi_{\mathrm{sh}}. \end{aligned}$$

Furthermore, $\Pi_{\mathrm{sh}}\Pi_\mu = \Pi_\mu - \Pi_\mu^2 - \Pi_{\mathrm{sc}}\Pi_\mu = 0$, and the same argument gives $\Pi_{\mathrm{sh}}\Pi_{\mathrm{sc}} = 0$. Symmetry gives the corresponding products in the reverse order.

We now decompose the output change as

$$u_\mu = \Pi_\mu u, \qquad u_{\mathrm{sc}} = \Pi_{\mathrm{sc}} u, \qquad u_{\mathrm{sh}} = \Pi_{\mathrm{sh}} u. \tag{75}$$

Since the projections sum to I_H,

$$u = u_\mu + u_{\mathrm{sc}} + u_{\mathrm{sh}}. \tag{76}$$

For distinct components j and k,

$$u_j^\top u_k = u^\top \Pi_j^\top \Pi_k u = 0.$$

Expanding the squared norm therefore gives

$$\|u\|_2^2 = \|u_\mu\|_2^2 + \|u_{\mathrm{sc}}\|_2^2 + \|u_{\mathrm{sh}}\|_2^2. \tag{77}$$

The three components have direct interpretations. Mean shift is $u_\mu = (\mu_o - \mu_b)\mathbf{1}$, which changes the overall level of the future window. Waveform scaling is $u_{\mathrm{sc}} = \kappa q$, where $\kappa = q^\top u / \|q\|_2^2$. It changes the coefficient of the centered base waveform from 1 to $1 + \kappa$. Residual shape correction is $u_{\mathrm{sh}} = u - u_\mu - u_{\mathrm{sc}}$. It has zero mean and is orthogonal to the centered base waveform. Here, residual refers to the change remaining after these two projections, and is distinct from the autoencoder reconstruction residual R_0.

When $q = 0$, the base forecast has no nonzero centered waveform direction. We then set $\Pi_{\mathrm{sc}} = 0$ and $\Pi_{\mathrm{sh}} = I_H - \Pi_\mu$. The decomposition and orthogonality identities remain valid. In implementation, we apply this rule whenever $H^{-1}\|q\|_2^2 \le 10^{-20}$. These channels remain in the evaluation, with their centered changes assigned to the shape component.

D.3.2 Component Errors and Contributions

The magnitude of a component does not determine whether it improves the forecast. We evaluate each component by comparing its prediction error before and after refinement. Let $\mathcal{K} = \{\mu, \mathrm{sc}, \mathrm{sh}\}$ and $e_k = \Pi_k e$ for $k \in \mathcal{K}$. The base and refined errors in component k are

$$\ell_k^{(0)} = \frac{1}{H}\|e_k\|_2^2, \qquad \ell_k^{(\mathrm{ref})} = \frac{1}{H}\|e_k - u_k\|_2^2. \tag{78}$$

The second expression follows from $y - o = e - u$ and the linearity of the fixed projection: $\Pi_k(y - o) = e_k - u_k$.

Orthogonality applies to both e and $e - u$. Consequently, their squared norms contain no cross terms between different components, giving

$$\begin{aligned} \mathrm{MSE}(b, y) &= \frac{1}{H}\|e\|_2^2 = \sum_{k \in \mathcal{K}} \ell_k^{(0)}, \\ \mathrm{MSE}(o, y) &= \frac{1}{H}\|e - u\|_2^2 = \sum_{k \in \mathcal{K}} \ell_k^{(\mathrm{ref})}. \end{aligned} \tag{79}$$

For a nondegenerate base waveform, these errors can also be written directly in terms of means, waveform coefficients, and residual shapes. For each $z \in \{b, o, y\}$, define

$$a_z = \frac{q^\top (z - \mu_z \mathbf{1})}{\|q\|_2^2}, \qquad \eta_z = z - \mu_z \mathbf{1} - a_z q. \tag{80}$$

By construction, $\mathbf{1}^\top \eta_z = 0$ and $q^\top \eta_z = 0$. In particular, $a_b = 1$ and $\eta_b = 0$. Subtracting the representations of o and y yields

$$o - y = (\mu_o - \mu_y)\mathbf{1} + (a_o - a_y)q + (\eta_o - \eta_y).$$

The three terms are orthogonal. Their squared norms therefore give

$$\mathrm{MSE}(o, y) = (\mu_o - \mu_y)^2 + (a_o - a_y)^2 \frac{\|q\|_2^2}{H} + \frac{\|\eta_o - \eta_y\|_2^2}{H}. \tag{81}$$

The corresponding base errors are $(\mu_b - \mu_y)^2$, $(1 - a_y)^2\|q\|_2^2/H$, and $\|\eta_y\|_2^2/H$. Thus, all three comparisons use the same base waveform as their reference.

We define the contribution of component k as its reduction in squared prediction error:

$$\begin{aligned} g_k &= \ell_k^{(0)} - \ell_k^{(\mathrm{ref})} \\ &= \frac{1}{H}\left(\|e_k\|_2^2 - \|e_k - u_k\|_2^2\right) \\ &= \frac{1}{H}\left[\|e_k\|_2^2 - \left(\|e_k\|_2^2 - 2e_k^\top u_k + \|u_k\|_2^2\right)\right] \\ &= \frac{2e_k^\top u_k - \|u_k\|_2^2}{H}. \end{aligned} \tag{82}$$

Since Π_k is symmetric and idempotent, $e_k^\top u_k = e^\top \Pi_k^\top \Pi_k u = e^\top u_k$. The contribution can therefore be calculated directly from the base error and the projected output change.

A positive g_k indicates error reduction, whereas a negative g_k indicates error increase. Equation 82 also explains why a larger change need not be beneficial. Improvement requires $2e_k^\top u_k > \|u_k\|_2^2$: alignment with the base error must outweigh the squared magnitude of the change.

Finally, subtracting the two identities in Equation 79 gives

$$\mathrm{MSE}(b, y) - \mathrm{MSE}(o, y) = \sum_{k \in \mathcal{K}} g_k. \tag{83}$$

This identity attributes the total MSE change to the three orthogonal output components. It does not require every component to have a positive contribution.

D.3.3 From State Refinement to Observable Forecast Changes

The decomposition above characterizes changes in the complete forecast output. To connect these observable changes to FUSE, recall that

$$A = D(S_{\mathrm{ref}}) - D(S_0), \qquad B = \beta R_0, \qquad Y_{\mathrm{ref}} - Y_0 = A + B. \tag{84}$$

Here, A denotes the decoded change induced by state refinement, while B denotes the adjustment of the reconstruction residual.

At future step t, the state path gives

$$A_t = D(s_{0,t} + \alpha \Delta s_t) - D(s_{0,t}). \tag{85}$$

Although α is shared across time, Δs_t varies with t, so the decoded correction can modify temporal structure rather than producing only a uniform shift or rescaling. Even with an affine decoder $D(s) = Ws + b_D$,

$$A_t = \alpha W \Delta s_t, \tag{86}$$

which remains time dependent through Δs_t.

For variable c, let

$$v = A_{:,c}, \qquad r_0 = (R_0)_{:,c}, \qquad u = o - b = v + \beta r_0, \tag{87}$$

where b, o, and u denote the base forecast, refined forecast, and their difference for that variable. Applying the fixed projections defined in Appendix D.3.1 yields

$$u_k = \Pi_k(v + \beta r_0) = \Pi_k v + \beta \Pi_k r_0, \qquad k \in \{\mu, \mathrm{sc}, \mathrm{sh}\}. \tag{88}$$

Thus, both the state-refinement path and the residual-adjustment path can contribute to each observable component.

For a nonzero centered base waveform q, the coordinates defined in Equation 80 satisfy

$$\begin{aligned} \mu_o - \mu_b &= \mu_v + \beta \mu_{r_0}, \\ a_o - 1 &= \frac{q^\top v}{\|q\|_2^2} + \beta \frac{q^\top r_0}{\|q\|_2^2}, \\ \eta_o &= \Pi_{\mathrm{sh}} v + \beta \Pi_{\mathrm{sh}} r_0. \end{aligned} \tag{89}$$

The first line follows by taking the mean of u. The second projects u onto the centered base waveform q, using $q^\top \mathbf{1} = 0$ and $a_b = 1$. The third follows from $\Pi_{\mathrm{sh}} b = 0$. Therefore, neither α nor β corresponds exclusively to mean, scale, or shape correction; the three components describe the resulting output change rather than separate internal mechanisms. When $q = 0$, Equation 88 remains valid under the projection convention defined above.

The target state can also be related to the final forecasting error. Let

$$S^\star = E(Y), \qquad R^\star = Y - D(S^\star). \tag{90}$$

Since $Y = D(S^\star) + R^\star$, we obtain

$$\begin{aligned} Y_{\mathrm{ref}} - Y &= D(S_{\mathrm{ref}}) + (1+\beta) R_0 - [D(S^\star) + R^\star] \\ &= D(S_{\mathrm{ref}}) - D(S^\star) + (1+\beta) R_0 - R^\star. \end{aligned} \tag{91}$$

Using the decoder Lipschitz constant K_D established in Appendix B.1, the triangle inequality gives

$$\begin{aligned} \|Y_{\mathrm{ref}} - Y\|_F &\le \|D(S_{\mathrm{ref}}) - D(S^\star)\|_F + \|(1+\beta) R_0 - R^\star\|_F \\ &\le K_D \|S_{\mathrm{ref}} - S^\star\|_F + \|(1+\beta) R_0 - R^\star\|_F. \end{aligned} \tag{92}$$

Hence, state distance controls only one term in an upper bound on forecasting error, while residual mismatch contributes independently. A smaller state distance therefore does not necessarily imply a smaller realized forecasting error. This motivates evaluating the component gains on the complete refined forecast rather than using state-space distance as a surrogate for output quality.

D.3.4 Aggregation Across Windows and Horizons

For a fixed dataset, backbone, and horizon H, let W_H be the number of test windows and C the number of variables. We construct the projections separately for each window and variable. Let $e_{n,c}$ and $g_{k,n,c}$ denote the base error and component contribution for window n and variable c. We first average over all test windows and variables:

$$\begin{aligned} \mathcal{E}_{0,H} &= \frac{1}{W_H C H} \sum_{n=1}^{W_H} \sum_{c=1}^{C} \|e_{n,c}\|_2^2, \\ \overline{g}_{k,H} &= \frac{1}{W_H C} \sum_{n=1}^{W_H} \sum_{c=1}^{C} g_{k,n,c}. \end{aligned} \tag{93}$$

Here, $\mathcal{E}_{0,H}$ is the total MSE of the base forecast. For $\mathcal{E}_{0,H} > 0$, the normalized contribution is

$$\mathrm{Contribution}_{k,H} = 100 \, \frac{\overline{g}_{k,H}}{\mathcal{E}_{0,H}}. \tag{94}$$

All three components use this same denominator. The normalization is applied after averaging errors over windows and variables, rather than averaging individual window ratios.

Let $\mathcal{E}_{\mathrm{ref},H}$ denote the refined forecast's total MSE under the same averaging. Averaging Equation 83 and dividing by $\mathcal{E}_{0,H}$ gives

$$\sum_{k \in \mathcal{K}} \mathrm{Contribution}_{k,H} = 100 \, \frac{\mathcal{E}_{0,H} - \mathcal{E}_{\mathrm{ref},H}}{\mathcal{E}_{0,H}}. \tag{95}$$

Thus, under these definitions, each component measures a contribution to the relative reduction in total MSE, rather than a reduction relative to that component's own error.

For Table 3, the analysis fixes $L = 336$. We calculate the normalized contributions separately for $H \in \mathcal{H} = \{96, 192, 336, 720\}$ and then take their arithmetic mean:

$$\overline{\text{Contribution}}_k = \frac{1}{|\mathcal{H}|} \sum_{H \in \mathcal{H}} \text{Contribution}_{k,H} \,. \tag{96}$$

Each horizon therefore receives equal weight, regardless of its number of test windows. These contributions concern MSE alone, whereas the Gain reported in the plug-in experiment combines relative MSE and MAE reductions.

D.3.5 Full Results of Forecast Refinement

Table 12 provides the horizon-wise contributions underlying Table 3. The analysis includes all test windows and channels at $L = 336$, following the definitions in Appendix D.3.

E Additional Ablations

E.1 Loss Ablation and Weight Sensitivity

We evaluate the state-supervision objective on ETTh1, ETTh2, ETTm1, and ETTm2 using the default MLP, $L = 336$, $H \in \{96, 720\}$. Within each setting, all variants share the same frozen base forecaster and autoencoder, predictor initialization, and zero-initialized α and β. Only the state-loss weights are changed, while the forecasting loss retains unit weight. Training uses AdamW (Loshchilov & Hutter, 2019), a constant learning rate of 5×10^{-4}, and batch size 512. The maximum budget is 120 epochs, with a minimum of 12 epochs, patience 12, and a relative validation improvement threshold of 10^{-5}. Checkpoints are selected by validation forecasting MSE, with the initial model included as a candidate and no lookback selection.

Loss ablation. We first isolate the roles of State MSE and State cosine supervision. Table 13 compares no state supervision, each loss individually, and their default joint objective. Relative test gains are computed for each evaluated configuration and then averaged equally. The joint objective achieves the largest mean test gain of 2.118%, followed by State MSE alone at 1.626%. Without state supervision, the mean gain is 1.147%, while State cosine alone yields 0.420%. These results indicate that State MSE provides the primary state supervision, while cosine supervision offers complementary benefit when combined with it. Because α is initialized at zero, state supervision also provides the predictor's direct training signal at initialization, as derived in Appendix B.3.

Loss-weight sensitivity. We further examine whether performance depends strongly on the exact choice of the two state-loss weights. We independently vary $\lambda_{\text{m}} \in \{5, 10, 20\}$ and $\lambda_{\text{c}} \in \{7.5, 15, 30\}$, producing a 3×3 grid around the default setting $(10, 15)$. Figure 6 reports the relative test MSE change with respect to the default for each ETT dataset. Each cell averages the corresponding horizons and seeds, with the second line showing the standard deviation. Most configurations remain close to the default, indicating limited sensitivity within the evaluated range. The variation is smallest on ETTh1 and ETTm2, while ETTh2 exhibits relatively larger sensitivity. Overall, the results show that the default $(10, 15)$ lies within a broad region of comparable performance rather than relying on a sharply tuned optimum.

E.2 Ablation of Learned State Refinement

We examine whether forecast gains depend on how a predicted state is used, rather than simply on having access to it. We freeze trained FUSE checkpoints and compare two uses of the same predicted state. Endpoint sets the state-refinement coefficient to one, whereas Learned step retains the scalar learned during FUSE training and shared across all test windows. Both set $\beta = 0$:

$$\begin{aligned} Y_{\text{end}} &= D(\widehat{S}) + R_0, \\ Y_{\text{step}} &= D(S_0 + \alpha \Delta S) + R_0. \end{aligned} \tag{97}$$

The two paths share the same test windows, predicted states, decoder, and reconstruction residual. They differ only in the movement coefficient, with no retraining.

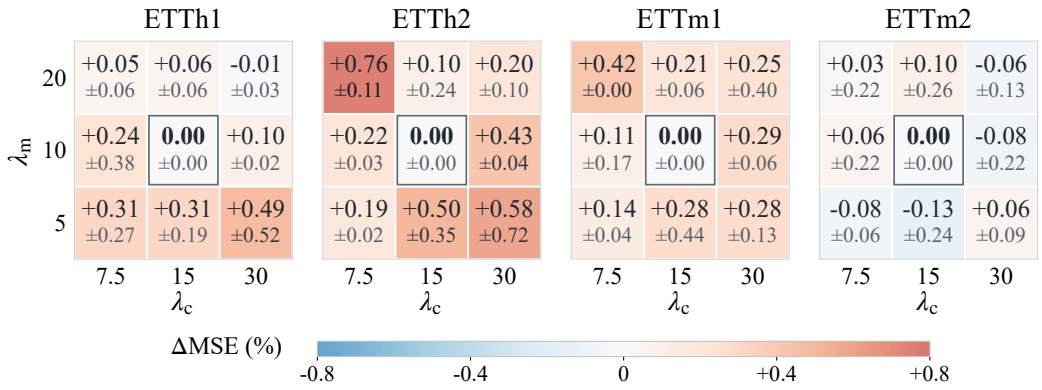

Figure 6: **Sensitivity to state-loss weights.** Relative test MSE change (%) with respect to the default $(\lambda_m, \lambda_c) = (10, 15)$ on the four ETT datasets; lower values are better. Each cell reports the mean change across $H \in \{96, 720\}$, with $\pm$ denoting the corresponding standard deviation. The default setting is outlined.

The study covers all eight datasets and $H \in \{96, 192, 336, 720\}$. We retain the lookback lengths used by the corresponding source experiments, with each paired comparison evaluated from the same selected checkpoint.

For setting j, let $\mathcal{E}_j(Z)$ denote the MSE of forecast Z over all test windows, future steps, and variables. Its gain relative to Base is

$$G_j(Z) = 100\,\frac{\mathcal{E}_j(Y_0) - \mathcal{E}_j(Z)}{\mathcal{E}_j(Y_0)}. \tag{98}$$

Table 14 averages these gains equally over horizons and seeds within each dataset. Mean Gain further averages the eight dataset gains.

Learned step yields higher mean gains in 21 of the 24 dataset–backbone pairs. For the default MLP, the overall mean gain changes from -2.733% to $+1.842\%$ without changing the predicted states. Endpoint performs better only for DLinear on ETTm1, ETTh2, and Solar, indicating that the learned step is not uniformly superior. Because each pair uses identical predicted states, the overall improvement reflects how those states are used rather than better state prediction. These results support treating predicted states as guidance with a learned movement coefficient rather than as fixed refinement endpoints.

Learned refinement coefficients. We inspect the learned coefficient α in FUSE checkpoints across all eight datasets and $H \in \{96, 192, 336, 720\}$, retaining the lookback lengths selected in the corresponding experiments. Table 15 summarizes the learned coefficients and their ranges across horizons and random seeds. Most trained models operate in the interpolation regime. For the default MLP on Traffic, all learned coefficients are negative, consistently yielding reversal across horizons and seeds. DLinear also enters the extrapolation regime in some checkpoints on Weather and Electricity.

Table 12: Contributions to MSE reduction (%) for each forecasting horizon. Mean, Scaling, and Shape denote Mean Shift, Waveform Scaling, and Residual Shape Correction, respectively. Each contribution is normalized by the corresponding base forecast's total MSE. Avg. averages the four horizons. Positive contributions are shown in **green bold** and negative contributions in gray. Colors and signs reflect unrounded values.

Dataset	H	Default MLP			DLinear			iTransformer		
		Mean	Scaling	Shape	Mean	Scaling	Shape	Mean	Scaling	Shape
ETTm1	96	**+0.96**	**+0.66**	**+0.58**	**+0.87**	**+3.77**	-0.01	**+4.11**	**+0.87**	**+3.17**
	192	**+1.23**	**+1.18**	**+0.45**	**+1.69**	**+2.36**	**+0.75**	**+4.01**	**+0.96**	**+2.73**
	336	**+1.27**	**+0.91**	**+0.43**	**+1.77**	**+1.84**	**+0.59**	**+2.56**	**+1.31**	**+3.02**
	720	**+1.44**	**+1.35**	**+0.21**	**+0.15**	**+0.09**	**+0.03**	**+3.60**	**+0.56**	**+2.82**
	Avg.	**+1.22**	**+1.03**	**+0.42**	**+1.12**	**+2.02**	**+0.34**	**+3.57**	**+0.93**	**+2.93**
ETTm2	96	**+0.83**	**+0.24**	**+0.42**	**+2.81**	**+0.53**	**+0.57**	**+3.97**	**+1.57**	**+3.90**
	192	**+1.39**	**+0.16**	**+0.43**	**+3.95**	**+0.43**	**+0.69**	**+1.50**	**+1.40**	**+3.06**
	336	**+0.48**	**+0.30**	**+0.48**	**+9.29**	**+1.19**	**+1.25**	**+1.61**	**+1.06**	**+2.87**
	720	**+0.95**	**+0.37**	**+0.60**	**+17.95**	**+1.47**	**+1.23**	**+1.33**	**+0.90**	**+2.37**
	Avg.	**+0.91**	**+0.27**	**+0.48**	**+8.50**	**+0.90**	**+0.93**	**+2.10**	**+1.23**	**+3.05**
ETTh1	96	**+0.97**	**+0.66**	**+0.25**	**+0.05**	-0.01	**+0.01**	**+3.25**	**+1.31**	**+2.40**
	192	**+0.38**	**+1.15**	**+0.72**	**+0.02**	**+0.00**	-0.00	**+1.70**	**+0.89**	**+3.01**
	336	**+0.07**	**+1.04**	**+0.72**	**+0.09**	**+0.02**	**+0.00**	**+1.97**	**+0.44**	**+2.60**
	720	-0.21	**+0.87**	**+0.81**	**+0.06**	**+0.02**	-0.00	-0.28	**+1.20**	**+5.40**
	Avg.	**+0.30**	**+0.93**	**+0.62**	**+0.05**	**+0.01**	**+0.00**	**+1.66**	**+0.96**	**+3.35**
ETTh2	96	**+2.08**	**+0.52**	**+0.79**	**+8.00**	**+0.60**	**+0.99**	**+3.82**	**+1.29**	**+3.69**
	192	-1.02	**+0.28**	**+1.24**	**+6.87**	**+1.11**	**+1.16**	-0.23	**+0.76**	**+3.51**
	336	**+0.47**	**+0.16**	**+0.56**	**+16.59**	**+2.17**	**+0.89**	**+8.25**	**+0.53**	**+2.34**
	720	**+2.35**	**+0.42**	**+1.09**	**+29.34**	**+5.66**	**+0.69**	**+2.00**	**+0.43**	**+0.99**
	Avg.	**+0.97**	**+0.34**	**+0.92**	**+15.20**	**+2.39**	**+0.93**	**+3.46**	**+0.76**	**+2.64**
Weather	96	**+0.46**	**+0.59**	**+0.18**	**+9.26**	**+1.84**	**+0.02**	**+6.47**	**+1.88**	**+0.44**
	192	-0.32	**+0.41**	**+0.22**	**+6.57**	**+1.21**	**+0.08**	**+2.72**	**+1.11**	**+0.66**
	336	**+0.32**	**+0.49**	**+0.23**	**+4.75**	**+1.22**	**+0.10**	**+3.25**	**+1.09**	**+0.52**
	720	**+1.33**	**+0.66**	-0.04	**+3.71**	**+0.90**	**+0.29**	**+1.97**	**+1.11**	**+0.35**
	Avg.	**+0.45**	**+0.54**	**+0.15**	**+6.07**	**+1.29**	**+0.12**	**+3.60**	**+1.30**	**+0.49**
Electricity	96	**+1.30**	**+0.61**	**+0.20**	**+1.59**	**+0.35**	**+1.13**	**+1.60**	**+0.59**	**+0.92**
	192	**+2.54**	**+0.55**	**+0.46**	-0.03	**+0.25**	**+1.90**	**+1.25**	**+0.44**	**+1.53**
	336	**+3.30**	**+0.44**	**+0.44**	**+0.16**	**+0.03**	**+1.52**	**+2.77**	**+0.73**	**+2.21**
	720	**+5.57**	**+0.45**	**+0.43**	**+0.47**	**+0.00**	**+1.68**	**+4.12**	**+0.53**	**+2.10**
	Avg.	**+3.18**	**+0.52**	**+0.38**	**+0.55**	**+0.16**	**+1.56**	**+2.43**	**+0.57**	**+1.69**
Traffic	96	**+1.35**	**+2.02**	**+0.02**	-4.51	**+4.59**	**+0.38**	**+3.31**	-4.40	**+1.04**
	192	**+1.38**	**+2.60**	-0.13	-5.29	**+5.67**	**+0.22**	**+3.84**	-5.86	**+1.09**
	336	**+1.02**	**+2.36**	**+0.02**	-4.82	**+5.13**	**+0.42**	**+3.20**	-6.01	**+0.85**
	720	**+0.68**	**+2.46**	**+0.03**	-4.00	**+3.84**	**+0.66**	**+2.01**	-4.65	**+0.41**
	Avg.	**+1.11**	**+2.36**	-0.02	-4.66	**+4.81**	**+0.42**	**+3.09**	-5.23	**+0.85**
Solar	96	**+3.42**	**+3.74**	**+0.98**	**+6.29**	**+6.68**	**+6.48**	**+4.97**	**+3.95**	**+2.39**
	192	**+0.72**	**+1.57**	**+0.56**	**+3.23**	**+6.36**	**+5.32**	**+0.72**	-0.15	**+2.01**
	336	**+0.15**	-0.04	**+0.26**	**+3.23**	**+8.66**	**+4.61**	**+0.95**	**+0.41**	**+2.35**
	720	**+0.29**	**+0.06**	**+0.26**	**+3.63**	**+8.78**	**+4.06**	**+0.39**	**+0.07**	**+2.01**
	Avg.	**+1.14**	**+1.33**	**+0.52**	**+4.09**	**+7.62**	**+5.12**	**+1.76**	**+1.07**	**+2.19**

Table 13: Loss ablation on the ETT datasets. The forecasting loss is retained in all variants. Test Gain denotes the mean relative test MSE reduction (%) against Base; higher is better. Best values are bold.

State supervision	λ_{m}	λ_{c}	Test Gain (%)
None	0	0	+1.147
State MSE	10	0	+1.626
State cosine	0	15	+0.420
Joint (default)	10	15	**+2.118**

Table 14: Frozen ablation of learned state refinement. Values are MSE reductions (%) relative to Base, with $\beta = 0$. Endpoint uses $\alpha = 1$; Learned step uses the trained α. Each dataset averages four horizons and the corresponding seeds. Bold indicates the better of Endpoint and Learned step within each pair.

	Default MLP		DLinear		iTransformer	
Dataset	Endpoint	Learned step	Endpoint	Learned step	Endpoint	Learned step
ETTm1	-0.622	**+1.877**	**+4.581**	+4.534	+6.752	**+7.442**
ETTm2	-3.772	**+0.193**	+2.445	**+4.633**	+4.667	**+6.804**
ETTh1	+0.641	**+2.460**	-7.375	**+0.568**	+4.380	**+4.689**
ETTh2	-2.155	**+1.417**	**+21.125**	+19.056	+6.534	**+6.678**
Weather	-5.890	**+0.284**	+6.667	**+7.041**	+5.198	**+6.662**
Electricity	-0.855	**+3.183**	+1.234	**+1.776**	+0.919	**+3.322**
Traffic	-10.410	**+0.137**	-4.626	**-0.008**	-14.651	**+0.282**
Solar	+1.203	**+5.185**	**+12.570**	+12.124	+3.471	**+5.537**
Mean Gain	-2.733	**+1.842**	+4.578	**+6.216**	+2.159	**+5.177**

Table 15: Learned state refinement coefficients. Each entry reports the mean α followed by its [minimum, maximum] across four horizons and the seeds.

Dataset	Default MLP	DLinear	iTransformer
ETTm1	$0.464\ [0.330, 0.593]$	$0.629\ [0.121, 0.881]$	$0.696\ [0.600, 0.837]$
ETTm2	$0.330\ [0.208, 0.459]$	$0.416\ [0.281, 0.569]$	$0.665\ [0.583, 0.746]$
ETTh1	$0.331\ [0.234, 0.459]$	$0.074\ [-0.010, 0.167]$	$0.391\ [0.191, 0.539]$
ETTh2	$0.294\ [0.160, 0.391]$	$0.478\ [0.329, 0.643]$	$0.515\ [0.235, 0.813]$
Weather	$0.426\ [0.223, 0.534]$	$0.935\ [0.748, 1.093]$	$0.598\ [0.510, 0.661]$
Electricity	$0.284\ [0.119, 0.475]$	$0.849\ [0.544, 1.096]$	$0.197\ [0.118, 0.256]$
Traffic	$-0.058\ [-0.103, -0.002]$	$-0.019\ [-0.037, 0.034]$	$0.059\ [0.014, 0.079]$
Solar	$0.523\ [0.361, 0.833]$	$0.815\ [0.668, 0.960]$	$0.685\ [0.498, 0.779]$