BEYOND PLUG-AND-PLAY: DO FORECASTING PLUGINS PAY OFF ACROSS MODELS AND SETTINGS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Forecasting plugins are increasingly proposed as plug-and-play or model-agnostic enhancements for time-series forecasting. However, architectural compatibility alone does not establish their effectiveness or practical deployment value across datasets, prediction horizons, and backbone models. We introduce PlugBench-TS, a controlled benchmark of ten forecasting enhancements spanning jointly fitted modules, training objectives, normalization, retrieval, output adaptation, and online forecast revision. It measures paired forecasting gain, checkpoint-pairing dependence where replacement is valid, and deployment memory and latency. The primary matrix covers 960 plugin–context combinations across four backbones, six datasets, and four prediction horizons, each evaluated in three paired runs. The results show why compatibility and a single accuracy number are not enough: the same enhancement can help in one context and hurt in another; a gain with its fitted backbone may disappear after a checkpoint change; and a small parameter count can conceal substantial retrieval memory and inference time. Building on these observations, we study whether the state used by online, static, and retrieval plugins can be reduced while preserving useful forecasting gains. The case studies reveal both successful reductions and accuracy–cost trade-offs. Overall, PlugBench-TS provides a reproducible basis for evaluating plugin value and for guiding future plugin design and deployment.

1 INTRODUCTION

Forecasting plugins aim to improve an existing model through a training loss, normalization step, retrieval memory, output correction, or online update. Some are designed to work with different backbones, yet an interface that accepts a backbone does not tell us whether forecasts improve across tasks or what the plugin costs to retain and run. We ask what value these plugins provide when each backbone uses a configuration based on its published implementation, without plugin-specific retuning of the bare reference.

PlugBench-TS compares a bare backbone with a plugin integration using the same backbone recipe, data splits, scaling, forecast origins, and seeds. We record where the plugin enters the model, whether it adds to or replaces a backbone component, what information it uses, and how it is fitted or updated. Each result thus refers to a specified plugin–backbone integration. Ordinary accuracy comparisons cannot tell us whether its fitted state can be reused after the backbone is retrained.

In the tested settings, the same plugin can help in one forecasting context and hurt in another (Figure 1). Fitted plugin state that helps with one checkpoint can lose its gain with another checkpoint of the same architecture. A plugin with few learned parameters can also require substantial retrieval memory and prediction time. The cost gap motivates a further question: can plugin-owned state be reduced while preserving the original integration and its useful forecasting gain?

The benchmark and follow-up study make three contributions:

1. **PlugBench-TS, a controlled plugin benchmark.** We evaluate ten plugins across four source-backed backbone families, six datasets, and four forecast horizons. Each plugin is paired with a bare reference, with its integration, information access, fitting, and update rules recorded.

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.

2. **Three measurements of plugin value.** We report context-level paired gains, same-architecture checkpoint replacements, and deployment state and latency. Targeted integration and source-recipe controls identify which tested setting each result describes.
3. **Cost reduction guided by the benchmark.** Our contract-matched plugin transformation procedure (CMPT) uses the recorded execution rules and resource costs to test smaller online adapters, static output maps, and retrieval indices. It checks whether each candidate preserves the original plugin's lifecycle and useful forecasting gain while reducing measured resources.

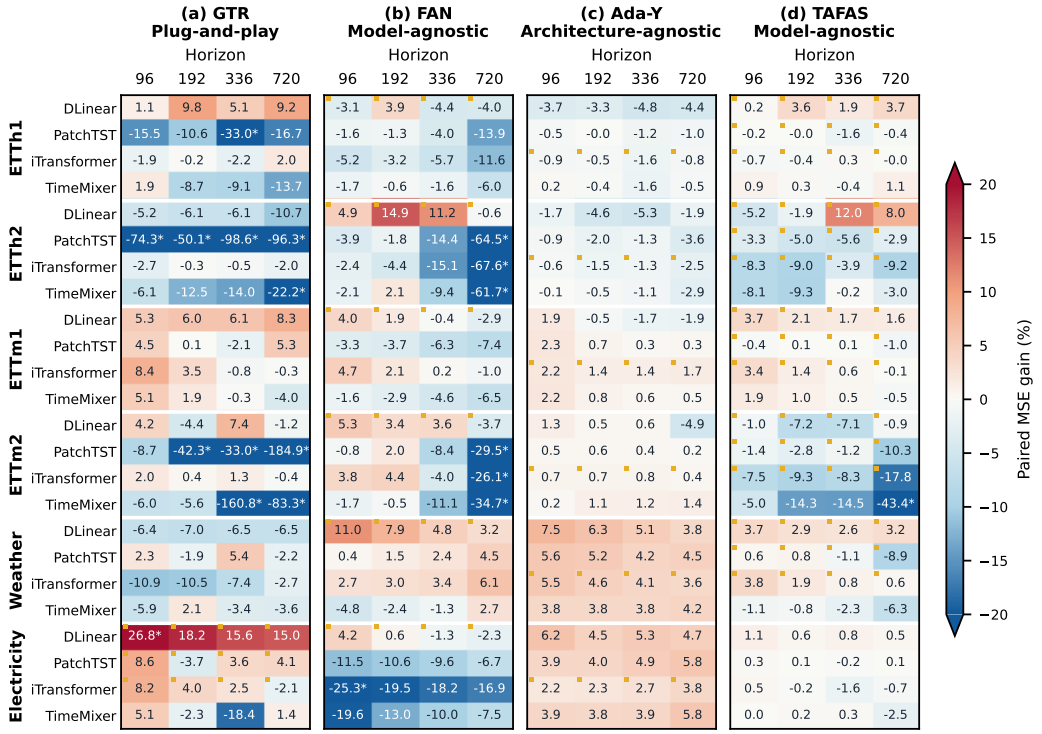

Figure 1: **Context-level gain for four plugin integrations** ($L = 96$, three matched seeds). Panel subtitles reflect the methods' stated compatibility claims. Each cell gives paired MSE gain after seed averaging; warm colors favor the backbone with the plugin. Amber corners mark source-reported dataset–backbone pairings evaluated under our shared protocol. Ada-Y is the output component of δ-Adapter; TAFAS follows its native online-revision protocol. Colors saturate at $\pm 20\%$, while numbers retain the observed values. Per-seed results appear in Appendix Table 6.

2 Related Work

Backbones and forecasting plugins. Our controlled suite spans four forecasting architectures: linear decomposition with DLinear, temporal patching with PatchTST, variate-token attention with iTransformer, and multiscale mixing with TimeMixer (Zeng et al., 2023; Nie et al., 2023; Liu et al., 2024; Wang et al., 2024b). The plugins we study act at different stages. FAN modifies frequency normalization (Ye et al., 2024), LatentTSF forecasts in a learned latent space (Yang et al., 2026), and FreDF modifies the training objective (Wang et al., 2025). GLAFF combines global calendar-derived and local information (Wang et al., 2024a), while GTR introduces learned global temporal references (Cao et al., 2026).

Retrieval, correction, and online adaptation. PIR identifies unreliable forecasts and revises them with local and global information (Liu et al., 2025). KUP-BI retrieves continuation-style information from a training library (Liu et al., 2026). δ-Adapter adjusts the input and output of a frozen

forecaster (Liang et al., 2026); we evaluate its Ada-Y output component. COSA and TAFAS adapt at deployment through different update and forecast-revision rules (Im & Kwon, 2026; Kim et al., 2025). We therefore document the tested integration of each published plugin, including extensions to unreported backbone pairings, and retain TAFAS's online-revision setting when comparing its forecasts.

Forecasting benchmarks. TSLib provides a common implementation library for time-series models (Wang et al., 2024c); TFB, BasicTS+, and TSPP standardize data and evaluation pipelines for forecasting comparisons (Qiu et al., 2024; Shao et al., 2023; Baczek et al., 2024). TimeRecipe and CombinationTS go further into architectural components, studying their effectiveness across tasks and model combinations (Zhao et al., 2026; Wang et al., 2026). These studies establish model- and component-level comparisons. PlugBench-TS asks a complementary question: what changes when a published plugin is fitted or attached to a specified backbone? We pair its forecasts with those of the same bare backbone, test dependence on the fitted checkpoint where replacement is meaningful, and examine deployment state and cost under the tested integration. Appendix B compares the evaluation scopes and lists the pinned implementations.

3 PLUGBENCH-TS: BENCHMARK DESIGN

PlugBench-TS records what a plugin may read, where it joins the backbone, how it is fitted, what it stores, and when it predicts or updates. These rules determine which comparisons are valid (Figure 2).

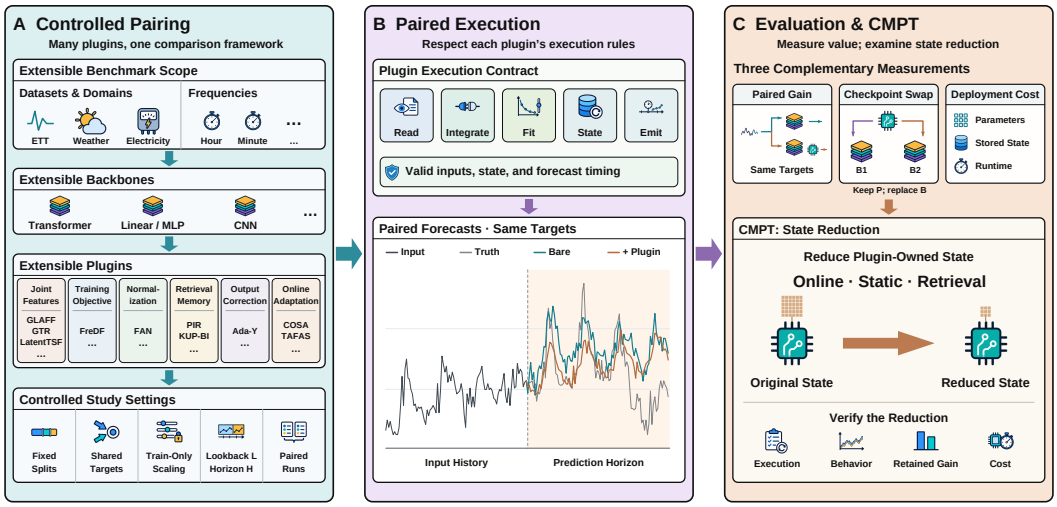

Figure 2: **PlugBench-TS workflow.** (A) Dataset–backbone–plugin pairings in an extensible suite. (B) Execution rules and paired bare-backbone and plugin forecasts on the same targets. (C) Paired gain, checkpoint replacement where applicable, and deployment cost guide the CMPT state-reduction studies.

3.1 EVALUATION CONTEXT AND EXECUTION CONTRACTS

An evaluation context is $c = (D, B, L, H, s, \pi_{\text{data}})$, where π_{data} fixes the shared split, scaling, forecast times, and metric. The benchmark compares a bare-backbone forecast $\hat{Y}_B$ with a forecast $\hat{Y}_P$ from the backbone using plugin P on the same targets. For jointly trained plugins, the measured difference includes their effect on backbone training. Table 1 lists the controlled forecasting settings. We also evaluate a longer input history ($L = 720$) to see whether the observed gains persist when the lookback changes; its coverage and results appear in Appendix E.2. An integration declares $\Gamma_P = (\mathsf{read}, \mathsf{integrate}, \mathsf{fit}, \mathsf{state}, \mathsf{emit})$: accessible variables and times; insertion site and add/replace policy; fitting relative to the backbone; persistent state and transitions; and output or revision timing. Thus c fixes the backbone and shared data protocol, whereas Γ_P describes how plu-

Table 1: Datasets, backbone models, forecast horizons, and plugin types in the controlled $L = 96$ matrix.

Item	Specification
Datasets	ETTh1, ETTh2, ETTm1, ETTm2, Weather, Electricity
Backbones	DLinear, PatchTST, iTransformer, TimeMixer
Forecast task	Supervised multivariate point forecasting; no pretrained zero-shot foundation models; $L = 96$; $H \in \{96, 192, 336, 720\}$
Repeats	Three paired runs per plugin–context combination (suite seeds 2021, 2022, 2023)
Non-revised forecast families	GLAFF, PIR, GTR, KUP-BI, Ada-Y, LatentTSF, FAN, FreDF
Online-revision cohort	COSA and TAFAS; forecasts revised after label release are not pooled with forecasts emitted once
Unit of comparison	Paired targets, origins, scaler and backbone configuration; lifecycle-specific training

gin P runs. Each result row requires both. A training-loss plugin acts through fit and has no active inference module; a normalization plugin may add to or replace a backbone component; retrieval and online adaptation retain or update state (Wang et al., 2025; Ye et al., 2024; Liu et al., 2026; Kim et al., 2025).

Before comparing forecasts, we check that each plugin was fitted and run according to its declared rules. Three measurements then address decisions that accuracy alone cannot settle. Paired-system gain compares the evaluated backbone–plugin integration with its matched bare backbone; for joint fitting, it includes changes to backbone training. A valid same-architecture checkpoint replacement tests whether the fitted plugin state can be kept when the backbone is retrained, avoiding a potentially costly refit. Resource measurements show what must be stored and computed to obtain the gain. We apply each measurement only where the plugin's integration makes it meaningful.

3.2 Controlled Protocol and Targeted Author-Setting Checks

The primary controlled matrix fixes $L = 96$ and $H \in \{96, 192, 336, 720\}$. ETT uses chronological boundaries corresponding to 12/4/4 months (8,640/2,880/2,880 hourly observations, multiplied by four for ETTm); Weather and Electricity use chronological 70/10/20 splits. Scaling is fitted on training data only. Variable order, dataset hashes, and forecast-origin identifiers are retained. Targets lie wholly inside their assigned split; their input histories may extend into the preceding split. Evaluation uses stride one and standardized-space MSE and MAE. Shared preprocessing and evaluation follow the comparability principle of forecasting benchmarks (Qiu et al., 2024; Shao et al., 2023); here the comparison is between a backbone and that backbone using a specified plugin. Controlled-matrix timing, memory, and OOM results are measured on 24-GB RTX 3090 GPUs.

We use DLinear, PatchTST, iTransformer, and TimeMixer to cover linear decomposition, temporal patching, variate-token attention, and multiscale mixing, respectively (Zeng et al., 2023; Nie et al., 2023; Liu et al., 2024; Wang et al., 2024b). The iTransformer and TimeMixer model classes come from a pinned TSLib snapshot; DLinear and PatchTST use pinned author-source implementations. Within each dataset–horizon–backbone context, the bare implementation and training recipe are fixed across plugin comparisons; architecture and optimizer settings follow pinned backbone-author scripts. This lets us test whether a plugin helps without tuning the bare reference separately for it. Plugin-owned objectives, options, fitting schedules, and integration mode are resolved from pinned author sources when available; unreported combinations are labeled extensions. Jointly fitted plugins update every declared trainable component; frozen and online plugins retain their stated rules. The measured gain belongs to this specified plugin–backbone pairing, not to an author-tuned source-paper system.

The matrix includes source-reported pairings and extensions to unreported datasets or backbones, always with the pinned backbone recipe. This distinguishes performance under a common comparison from results obtained after tuning a backbone for a particular plugin. Integration details can matter: our controlled GTR–PatchTST pairing retains PatchTST's RevIN, whereas the released GTR configuration disables it; FAN is added to some backbones that already normalize their inputs.

A separate GTR author-recipe run with RevIN disabled and FAN replacement controls document these settings without changing the controlled cells (Appendix B).

New plugins can be registered through a model adapter and configuration, then compared with their bare backbones under a specified dataset, lookback, horizon, and backbone recipe (Appendix B).

3.3 Evaluation Questions and Measurements

We formalize the execution check and three measurements introduced above. Their assumptions and limits appear in Appendix C.

Execution validity: is the declared method actually being evaluated? A run is admitted only when its realized trace obeys $\Gamma_P = (\mathsf{read}, \mathsf{integrate}, \mathsf{fit}, \mathsf{state}, \mathsf{emit})$: variables are available when read; insertion, replacement, and disabled components match the registered integration; fitting and frozen/trainable flags match; state follows its schedule; and scored predictions follow the emission rule. Paired comparisons additionally hold c fixed, including targets, origins, and scaling. Source-paper reproduction further requires a matching author recipe and provenance.

Paired payoff: does the evaluated integration improve prediction? Fix a context c and write $R_c(f) = \mathbb{E}_c\|Y - f(X)\|^2$. For the bare backbone B and its paired forecast using plugin P, $G_c = 100[R_c(B) - R_c(P)]/R_c(B)$. This is a system-level difference when P also changes backbone fitting. For an output correction $a(Z)$ that uses only deployable information, with $e = Y - \hat{Y}_B$ and $m(Z) = \mathbb{E}[e \mid Z]$,

$$R_c(B) - R_c(B + a) = \mathbb{E}\|m(Z)\|^2 - \mathbb{E}\|a(Z) - m(Z)\|^2. \tag{1}$$

The identity characterizes output correction; it does not decompose a jointly trained system's gain. We measure paired gains; fitted probes do not identify the population optimum m. Context profiles stratify these gains by dataset, backbone, horizon, and lookback, but do not create independent replications.

Checkpoint-pairing dependence: does learned state survive checkpoint replacement? A deployed backbone may be retrained while its fitted plugin state is retained. If that state works only with the original checkpoint, the plugin must be refitted after each replacement. Where the operation is valid, we therefore evaluate A: independent bare backbone; B: paired backbone with the plugin; C: paired backbone with the plugin disabled; and D: the learned plugin attached to an independently trained checkpoint of the same architecture. With $\Delta_{\text{full}} = R(A) - R(B)$, $\Delta_{\text{back}} = R(A) - R(C)$, and $\Delta_{\text{trans}} = R(A) - R(D)$,

$$\Delta_{\text{full}} = \Delta_{\text{back}} + \Delta_{\text{trans}} + I, \quad I = [R(C) - R(B)] - [R(A) - R(D)]. \tag{2}$$

For every admitted replacement, the current matrix records all four systems: the disabled C arm is exported as `joint_only`. Representative production-path round trips further verify that extracting and reattaching the plugin to its own paired checkpoint preserves predictions (Appendix Table 18). Equation (2) can therefore be evaluated as an operational interaction contrast (Appendix Table 19); it is not a causal allocation. A failed replacement indicates dependence on the original pairing under this operation, not failure of the paired method or an obligation that every jointly trained method be portable. Appendix C.2 gives a local quadratic explanation, while successful replacement alone does not establish representational equivalence (Smith et al., 2025).

Deployment burden: which quality–resource trade-offs are feasible? For workload and hardware w, record $\mathbf{C}_w(f) = (C_{\text{offline}}, T_{\text{online}}, M_{\text{peak}}, S_{\text{deploy}})$, with learned parameters and persistent state itemized. The ideal constrained frontier is

$$R^*(\mathbf{b}, \mathcal{I}) = \inf_{f \in \mathcal{F}(\mathcal{I}):\, \mathbf{C}_w(f) \le \mathbf{b}} R(f), \tag{3}$$

where $\mathcal{I}$ specifies permitted information. We measure only evaluated implementations, keeping offline preparation, online updates, persistent state, and forward latency separate.

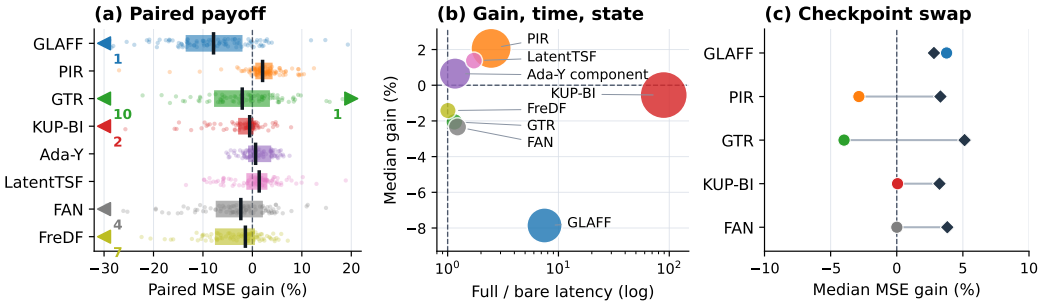

Figure 3: **Plugin payoff, cost, and checkpoint dependence.** (a) Three-run paired gains for 768 forecast-once contexts. (b) Gain versus clean RTX 3090 latency; bubble area encodes persistent plugin state. KUP-BI uses the on-demand port. (c) Median gain before (diamonds) and after (circles) checkpoint replacement, for 153 evaluated pairings with observed gain $\geq 1\%$. This test-selected view is descriptive; Table 3 reports all 480 replacements. All panels use the primary $L = 96$ setting.

Table 2: Paired MSE gain for eight forecast-once integrations at $L = 96$ (three runs per context). Source-scope denotes a reported plugin recipe and pairing under the controlled data and backbone settings; extension is an unreported pairing. GLAFF and Ada-Y rows name their evaluated variants. Rows describe specified integrations with different plugin recipes, not a method ranking.

Method	Scope	Positive/Done	Median (%)	IQR (%)
GLAFF	Paper-snapshot controlled	16/96	-7.86	[-13.50, -2.00]
PIR	Source-scope	59/67	+1.62	[+0.69, +3.16]
	Extension	21/29	+3.97	[-0.32, +6.78]
GTR	Source-scope	10/12	+6.14	[+3.36, +15.14]
	Extension	27/84	-2.47	[-9.48, +1.86]
KUP-BI	Extension	32/96	-0.55	[-2.88, +0.24]
Ada-Y component	Ada-Y output only	61/96	+0.65	[-0.87, +3.81]
LatentTSF	Extension	65/96	+1.38	[-1.23, +3.04]
FAN	Source-scope	15/24	+3.28	[-1.52, +4.81]
	Extension	16/72	-3.95	[-10.72, -0.77]
FreDF	Source-scope	25/64	-0.74	[-3.35, +1.50]
	Extension	3/32	-7.24	[-13.30, -3.38]

4 Benchmark Results and Analysis

Scope of the results. The bare backbone and its plugin integration use the same test windows, targets, preprocessing, and permitted label history. At $L = 96$, all ten plugins are evaluated over six datasets, four backbones, four horizons, and three paired runs per context. Each context's gain is computed from its mean paired bare and enhanced MSEs. Backbone recipes follow pinned official sources; plugin pairings beyond source-reported settings are identified as extensions (Appendix B). A separate $L = 720$ track tests a longer input history without pooling its results into this matrix.

4.1 Payoff and Protocol Scope

We first ask whether each evaluated plugin–backbone integration improves forecasts relative to its matched bare backbone. Figure 1 illustrates the variation across four plugin types; Table 2 summarizes all eight integrations that emit once. The source-scope and extension strata are reported separately in Appendix Table 15. An *extension* moves a plugin beyond its source-reported pairing or lookback under the benchmark's backbone recipe. These cells test whether the specified integration remains useful at the new operating point; source-reported settings are assessed separately. GTR improves 10/12 source-scope pairings but 27/84 extensions; FAN improves 15/24 source-scope DLinear pairings but 16/72 stacking extensions. These groups contain different contexts and do not isolate a cause. The GLAFF row uses its paper-snapshot architecture with train-only scaling; Ada-Y is the output component of δ-Adapter. These comparisons measure paired systems, not a module's

isolated effect. FAN controls show that improvement over a no-normalization backbone need not beat the canonical normalized backbone (Appendix Tables 9–11).

Selected author-runner checks for KUP-BI, FreDF, and LatentTSF recover source-reported trends and approximate MSEs (Appendix Table 13). All ten methods have three paired runs per $L = 96$ context. Appendix Figures 4–6 show the other six methods; Appendix Tables 6 and 26 give context-level values.

The separate $L = 720$ track appears in Appendix E.2; native online revisions appear in Appendix B. A same-protocol ETTh2/DLinear check reverses GLAFF's gain sign between its paper snapshot and current HEAD (Appendix F).

4.2 Checkpoint-pairing dependence

A deployment team may retrain a bare backbone and wish to reuse an already fitted plugin. We test that operation by attaching the saved plugin state to an independently trained bare checkpoint of the same architecture, where the registered integration permits it. This main replacement test is a reuse stress test, not an exchange between two jointly trained enhanced systems.

Table 3: Fitted plugin state attached to each cell's independently trained bare checkpoint of the same architecture over three seeds. The subset is selected by observed full-system test gain $\geq 1\%$; GLAFF uses the paper-snapshot version.

	All valid contexts			Observed full gain $\geq 1\%$			
Method	n	Trans. med.	Trans. pos.	n	Full med.	Trans. med.	Trans. pos.
GLAFF	96	-5.24	19/96	13	+2.81	+3.77	13/13
PIR	96	-2.61	14/96	64	+3.30	-2.86	9/64
GTR	96	-23.00	12/96	35	+5.13	-3.98	9/35
KUP-BI	96	+0.03	62/96	13	+3.23	+0.06	7/13
FAN	96	-7.14	15/96	28	+3.82	+0.01	14/28

Table 3 reports all 480 valid same-architecture replacements using three seeds per context. Figure 3(c) focuses on the 153 evaluated pairings with at least 1% observed full-system test gain. In this descriptive subset, GTR's median gain changes from $+5.13\%$ before replacement to -3.98% afterward, with 9/35 replacements still better than bare; PIR also falls from a $+3.30\%$ source median to -2.86% after replacement. Outcomes differ across integrations: GLAFF remains positive in all 13 selected replacements, while FAN and KUP-BI have replacement medians close to zero. Four production-path self-round-trips reproduce the original paired predictions exactly, and all admitted replacements include a disabled-plugin control (Appendix Tables 18 and 19). These outcomes describe how the fitted state behaves with a different checkpoint under the tested integrations; the favorable subset was selected using observed test gain and does not estimate a prospective deployment success rate. A separate three-seed exchange between jointly trained systems found that FAN stayed close to its native pairing, PIR lost some accuracy, and GTR could still deteriorate substantially in the tested extensions, so joint training of the target alone does not ensure plugin-state reuse (Appendix Table 20).

4.3 Deployment burden

Checkpoint portability is only one deployment concern. A plugin that remains paired with its source backbone may still require substantial retrieval, update, or storage resources, so parameter count and measured execution must be recorded separately.

Table 4: Deployment measurements for eight forecast-once integrations at $L = 96$. Each has 96 same-source seed-2021 RTX 3090 clean timing cells; forecast gains use three paired runs. Latency is the median cellwise full/bare ratio [IQR]. Extra parameters and learned tensor bytes are net differences; retrieval state is separate. KUP-BI uses the on-demand streaming port.

Method	Extra params	Learned MiB	Latency ratio [IQR]	External state
GLAFF	6,515,721	24.856	7.49 [3.76, 24.35]	–
PIR	362,202	1.382	2.47 [1.60, 5.17]	90.43 MiB [20.52 MiB, 2.15 GiB], device
GTR	10,034	0.038	1.15 [1.07, 1.25]	–
KUP-BI	128	< 0.001	89.11 [33.87, 211.92]	701.24 MiB [158.92 MiB, 16.69 GiB], host
Ada-Y component	2,673,968	10.200	1.17 [1.07, 1.43]	–
LatentTSF	18,503	0.071	1.72 [1.17, 1.91]	–
FAN	61,200	0.233	1.23 [1.12, 1.48]	–
FreDF	0	0	1.01 [0.99, 1.03]	–

Clean replay shows why learned parameter count is an incomplete cost measure (Table 4). Our on-demand KUP-BI port adds a median 128 learned parameters yet retains a median 701.24 MiB host index and has an $89.11\times$ full/bare latency ratio. The author's cached runner prepares retrieval earlier; this ratio describes our port, not every KUP-BI deployment (Appendix G). PIR's median device retrieval state is 90.43 MiB, while FreDF adds no deployment parameters and has a $1.01\times$ ratio. The $L = 720$ resource outcomes appear in Appendix E.2.

4.4 From deployment measurement to resource reduction

Parameter count misses what some plugins actually store and compute. The benchmark's execution records identify online adapter state, static output maps, and retrieval indices as distinct targets for reduction. The payoff and checkpoint results also tell us to test a smaller implementation in its original pairing rather than assume that lower cost preserves its gain. Section 5 turns these observations into a transformation test.

5 Using the Benchmark to Reduce Plugin Cost

5.1 What must remain unchanged?

The benchmark's state inventory and cost measurements point to different resource targets. CMPT turns them into a type-specific transformation test: within the original backbone–plugin pairing, a candidate must retain the same information access, integration, fitting, updates, and prediction timing. We then measure its behavior, forecast error, retained gain, and resource savings separately. The online, static, and retrieval cases use different transformations under these shared checks.

For state type τ, let $T_\tau(P; r) = P'$ denote a transformation with resource setting r. It is admissible at operating point ω only if

$$\begin{aligned} \Gamma_\omega^{\text{life}}(P') &\equiv \Gamma_\omega^{\text{life}}(P), \qquad \Gamma = (\mathsf{read}, \mathsf{integrate}, \mathsf{fit}, \mathsf{state}, \mathsf{emit}), \\ \mathcal{V}_\omega(P', P) &= (D_V, \Delta_R, \Delta\mathbf{C}), \quad D_V = \frac{1}{|V|HC} \sum_{i \in V} \|\hat{Y}_i^{P'} - \hat{Y}_i^{P}\|_F^2, \end{aligned} \tag{4}$$

where $\Delta_R = 100[R(P') - R(P)]/R(P)$ compares the reduced plugin with the original plugin, not with bare. Lifecycle equivalence preserves admissible reads and availability, integration role, sample and label-release rules, update and emission schedules, and the functional role of state. Parameterization and predictions may change; D_V, held-out Δ_R, and $\Delta\mathbf{C}$ measure the consequences. Deleting the plugin, changing an add operation into replacement, or replacing online updates with a static map changes the contract.

Behavioral proximity alone does not measure retained value. For a useful source, $R(P) < R(B)$, define

$$Q_{\text{ret}} = \frac{R(B) - R(P')}{R(B) - R(P)}. \tag{5}$$

We report the ratio only when source gain reaches 1%, avoiding unstable denominators. Contract, behavior, payoff, and cost are checked separately; per-cell results and bounds are in Appendix C.4.

5.2 Evidence across three kinds of state

We test COSA's online adapter, Ada-Y's static correction, and PIR's retrieval index. CPLR factorizes a fixed-batch COSA reference ($B = 48$), rather than the benchmark's PAAS run, while retaining that reference's causal update and output schedule. Low-rank Ada-Y changes only its output map. CPLM projects PIR's train-only keys while retaining memory eligibility, retrieval timing, k, and correction logic.

The same state reduction can save resources yet lose the original integration's useful forecast gain. To study whether reducing state preserves an actual benefit, we assess retained gain in cells where that integration improves on bare by at least 1%; we still report the other cells. Table 5 compares original and reduced gains at fixed backbone pairings, alongside retained gain and state size. Absolute test MSE for all three systems appears in Appendix Table 17.

Table 5: Forecast value retained in three plugin-state reductions at fixed backbone pairings. CPLR, Ada-Y, and PIR cover 8, 8, and 12 contexts, respectively, each evaluated with three seeds. "Seed runs" and "Useful" count context–seed pairs; the latter have observed source gain $\geq 1\%$ over bare. Gains, retained payoff, and state reduction are medians over the same useful runs; positive counts show when the reduced plugin still beats bare. The CPLR row combines six studied and two additional contexts; a separate two-backbone feasibility check without useful source gain appears in Appendix C.4. State ratios exclude runtime.

Transformation	Seed runs	Useful n	Source gain	Reduced gain	Positive	Med. retained	Med. state ↓
CPLR	24	14	+2.36	+2.33	14/14	0.92	91.6× params
Low-rank Ada-Y	24	13	+5.74	+3.02	12/13	0.49	30.3× params
PIR key projection	36	31	+4.11	+3.59	25/31	0.79	2.8× index

Among useful-source cells, CPLR remains beneficial in 14/14 and retains a median 0.92 of the original gain; low-rank Ada-Y does so in 12/13 (0.49), and PIR key projection in 25/31 (0.79). These are different resource–accuracy trades at fixed pairings. A second retrieval case rearranges the index of our on-demand KUP-BI port: across 12 contexts it preserves test forecasts bitwise, makes persistent host-index arrays $1.328\times$ smaller, and lowers median full-system validation-batch latency to $0.274\times$ that of the original port (Appendix C.4). The registered $+1\%$ source-relative MSE tolerance and its failures are reported by context in Appendix C.4; it does not mean 99% of the original gain is retained.

6 Discussion and Conclusion

The main matrix covers six datasets from three sources—ETT, Weather, and Electricity—and four backbone families. We also evaluate Traffic as a separate extension; more distinct datasets and backbones are needed to test how widely these findings hold. The joint-to-joint follow-up covers three plugins and two datasets and does not identify why replacement losses occur. Process RSS and serialization are measured on representative retrieval cells. Probabilistic forecasts, pretrained foundation models, and other hardware remain untested.

Future work should test whether plugins deliver gains across more datasets and backbone families without per-pair backbone retuning, while recording any required normalization replacement or joint fitting. Broader joint-to-joint swaps in source-reported settings could further test checkpoint portability. Extending the evaluation to probabilistic forecasts and measuring host-process peaks and serialization across the matrix would broaden its deployment scope.

PlugBench-TS measures what happens when a forecasting plugin is used with a specified backbone. Its controlled matrix tests paired gains at a common backbone operating point; targeted author-setting checks identify how selected released configurations behave under their own settings. In the evaluated pairings, plugin benefit varies with context and integration, fitted state can depend on its backbone checkpoint, and learned parameter count can understate deployment cost. The same integration, payoff, and cost records guide CMPT's tests of smaller online, static, and retrieval implementations in their original pairings. Improving gain stability and checkpoint portability remains open.

REPRODUCIBILITY STATEMENT

PlugBench-TS records resolved configurations and provenance during execution. Section 3 specifies the controlled data, backbone, selection, execution-contract, and measurement protocols. Appendix B documents evaluated integration identities, pinned source provenance, recipe resolution, data processing, information access, executable validity checks, and online event ordering. Appendix E reports coverage and hardware-scoped resource outcomes, Appendix G defines timing and deployment-cost boundaries, and Appendix C states the assumptions behind the diagnostic equations. An anonymous supplementary ZIP containing source code and compact numerical results accompanies this submission. It includes the benchmark and case-study code, locked suite/shard configurations, a pinned source catalog, per-seed result ledgers with configuration and source-hash signatures, coverage and cost records, and a script to redraw numerical heatmaps and diagnostic panels. The attachment omits raw datasets, checkpoints, and training logs; dataset acquisition and processing are documented in Appendix B, and pinned retrieval instructions are supplied for upstream components that we cannot redistribute. We plan to release the code in a public repository after review.

AI USE STATEMENT

We used generative AI tools to help formulate and critique the evaluation design and equations, draft and check the appendix proofs, plan experiments, implement and debug code, analyze recorded results, discover related work, prototype figures, and draft and edit manuscript text. AI-assisted code was tested, and numerical claims were compared with saved experimental records. We take responsibility for the final paper and supplementary artifacts, including AI-assisted content.

REFERENCES

Jan Baczek, Dmytro Zhylko, Gilberto Titericz, Sajad Darabi, Jean-Francois Puget, Izzy Putterman, Dawid Majchrowski, Anmol Gupta, Kyle Kranen, and Pawel Morkisz. TSPP: A unified benchmarking tool for time-series forecasting, 2024. URL https://arxiv.org/abs/2312.17100.

Fanpu Cao, Lu Dai, Jindong Han, and Hui Xiong. Enhancing multivariate time series forecasting with global temporal retrieval. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7a3284ce53f620d244fca0c79d94c478-Abstract-Conference.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.

HyunGi Kim, Siwon Kim, Jisoo Mok, and Sungroh Yoon. Battling the non-stationarity in time series forecasting via test-time adaptation. arXiv preprint, 2025. URL https://arxiv.org/abs/2501.04970.

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.

Chong Liu, Yingjie Zhou, Hao Li, Pengyang Wang, Qingsong Wen, and Ce Zhu. Beyond extrapolation: Knowledge utilization paradigm with bidirectional inspiration for time series forecasting. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.19249.

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://proceedings.iclr.cc/paper_files/paper/2024/hash/2ea18fdc667e0ef2ad82b2b4d65147ad-Abstract-Conference.html.

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*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/331c41353b053683e17f7c88a797701d-Abstract-Conference.html.

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://iclr.cc/virtual/2023/poster/10876.

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, 2024. URL https://arxiv.org/abs/2403.20150.

Zezhi Shao, Fei Wang, Yongjun Xu, Wei Wei, Chengqing Yu, Zhao Zhang, Di Yao, Tao Sun, Guangyin Jin, Xin Cao, Gao Cong, Christian S. Jensen, and Xueqi Cheng. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis, 2023. URL https://arxiv.org/abs/2310.06119.

Damian Smith, Harvey Mannering, and Antonia Marcu. Functional alignment can mislead: Examining model stitching. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55972–55998, 2025. URL https://proceedings.mlr.press/v267/smith25a.html.

Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, and Jianxin Liao. Rethinking the power of timestamps for robust time series forecasting: A global-local fusion perspective. In *Advances in Neural Information Processing Systems*, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2783192ea2696ee2ceb8746f5eea6681-Abstract-Conference.html.

Hao Wang, Lichen Pan, Yuan Shen, Zhichao Chen, Degui Yang, Yifei Yang, Sen Zhang, Xinggao Liu, Haoxuan Li, and Dacheng Tao. FreDF: Learning to forecast in the frequency domain. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1457fb1e5d72cdc4ecd88bc10f916095-Abstract-Conference.html.

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

Xiaorui Wang, Fanda Fan, Chenxi Wang, Yuxuan Yang, Rui Tang, Kuoyu Gao, Simiao Pang, Yuanfeng Shang, Zhipeng Liu, Wanling Gao, Lei Wang, and Jianfeng Zhan. CombinationTS: A modular framework for understanding time-series forecasting models. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=CwHRT46VmC.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Chen Wang, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark, 2024c. URL https://arxiv.org/abs/2407.13278.

Jie Yang, Yifan Hu, Yuante Li, Kexin Zhang, Kaize Ding, and Philip S. Yu. From observations to states: Latent time series forecasting. arXiv preprint, 2026. URL https://arxiv.org/abs/2602.00297.

Weiwei Ye, Songgaojun Deng, Qiaosha Zou, and Ning Gui. Frequency adaptive normalization for non-stationary time series forecasting. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/37c6d0bc4d2917dcbea693b18504bd87-Abstract-Conference.html.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/26317.

Zhiyuan Zhao, Juntong Ni, Shangqing Xu, Haoxin Liu, Wei Jin, and B. Aditya Prakash. TimeRecipe: A time-series forecasting recipe via benchmarking module level effectiveness. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/f46ddea413df86832418c5e04e59644f-Abstract-Conference.html.

A THREE-SEED CONTEXT RESULTS FOR FIGURE 1

Figure 1 highlights four plugins spanning joint fitting (GTR), normalization (FAN), a frozen output component (Ada-Y), and native online revision (TAFAS). Each displayed context uses seeds 2021–2023. Table 6 reports their backbone-level gain and seed dispersion; the underlying per-seed MSEs are in `data/figure1_multiseed_per_seed.csv`. The remaining six plugins have the same three-seed coverage in the complete matrix reported below.

Table 6: Multi-seed summary for the four methods in Figure 1 (3 seeds: 2021, 2022, 2023). For each context, bare and enhanced MSE are averaged across matched seeds before computing gain. Backbone and overall entries report positive contexts/completed contexts (median context-level gain, %). Stable $+/-$ counts contexts that are positive in all three seeds / non-positive in all three; Seed SD is the median within-context sample standard deviation of paired gain in percentage points. TAFAS retains its separate native online-revision contract.

Method	DLinear	PatchTST	iTransformer	TimeMixer	Overall	Stable $+/-$	Seed SD
GTR	14/24 (+4.67)	8/24 (-6.20)	9/24 (-0.32)	6/24 (-5.78)	37/96 (-2.05)	24/36	2.86
FAN	15/24 (+3.28)	5/24 (-3.95)	9/24 (-3.61)	2/24 (-3.75)	31/96 (-2.34)	27/59	0.78
Ada-Y (δ-Adapter)	12/24 (-0.00)	16/24 (+0.47)	16/24 (+1.06)	17/24 (+0.96)	61/96 (+0.65)	53/31	0.32
TAFAS	18/24 (+1.67)	7/24 (-0.72)	9/24 (-0.31)	9/24 (-0.63)	43/96 (-0.18)	31/40	0.62

The figure's per-context bare and plugin MSEs, gains, and across-seed dispersion are retained in `data/figure1_multiseed_per_seed.csv`. The three forecast-once integrations and native TAFAS share a display for coverage only; their forecasting protocols remain distinct.

A.1 $L = 96$ HEATMAPS FOR THE OTHER SIX PLUGINS

Figures 4–6 show the six plugins not pictured in Figure 1. All 96 dataset–backbone–horizon contexts per method have three accepted paired runs. Gain is computed from the across-run mean bare and enhanced MSEs within each plugin; the native COSA revised-output results remain separate from forecast-once methods. Colors saturate at $\pm 20\%$, while an asterisk flags a printed value beyond the display scale. GLAFF uses the paper-snapshot plugin recipe under the controlled data protocol.

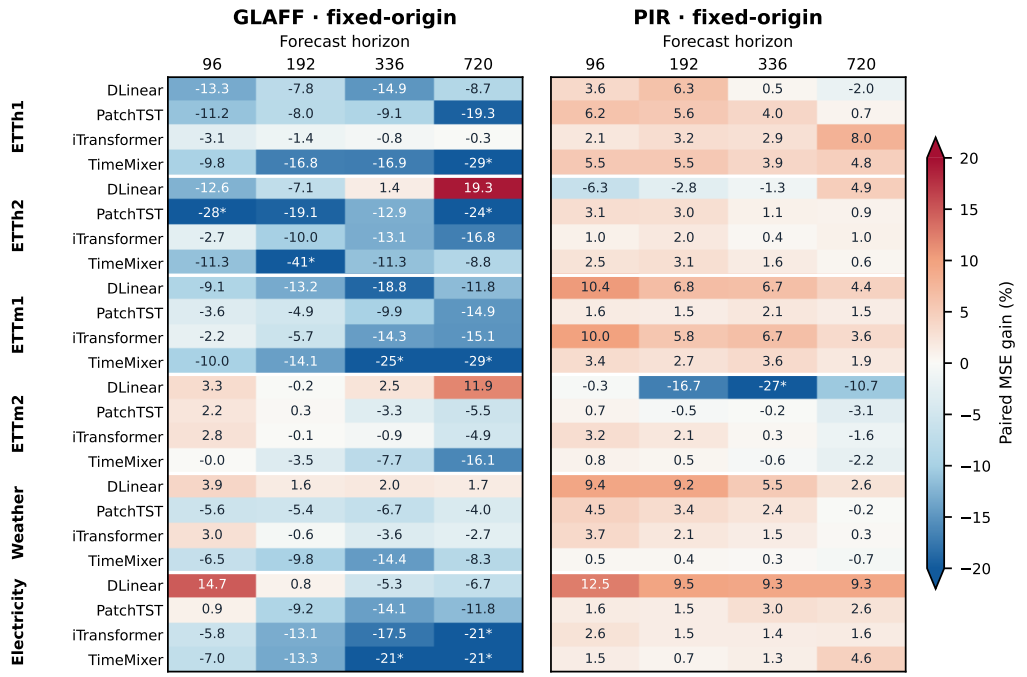

Figure 4: Three-run $L = 96$ paired MSE gains for GLAFF and PIR. Rows identify datasets and backbones; columns identify prediction horizons. GLAFF uses the controlled paper-snapshot recipe.

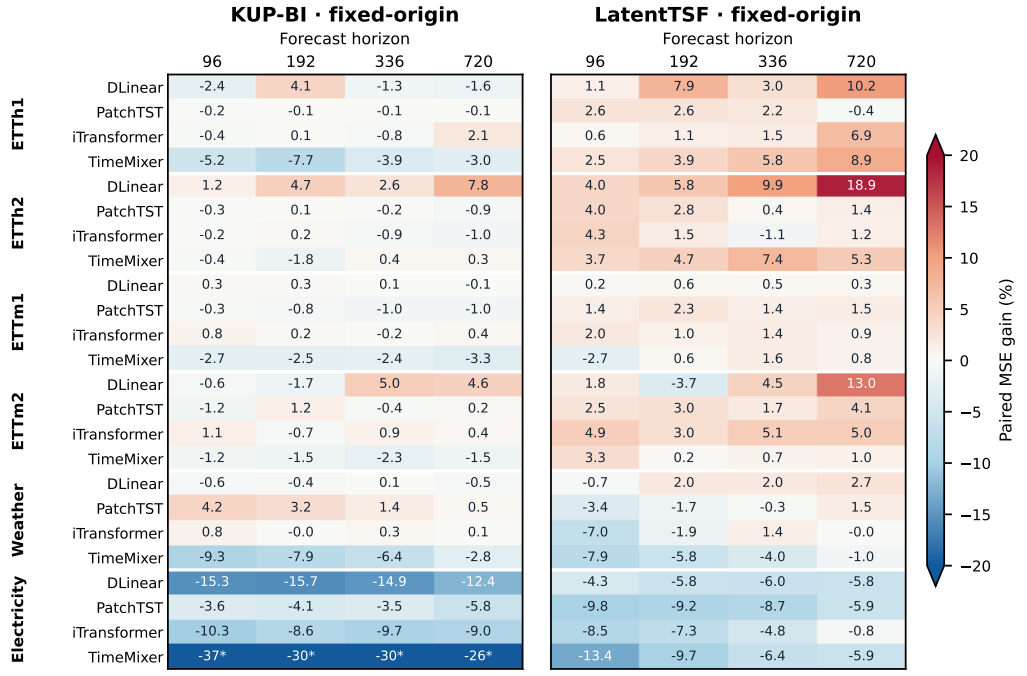

Figure 5: Three-run $L = 96$ paired MSE gains for KUP-BI and LatentTSF, using the scale of Figure 4.

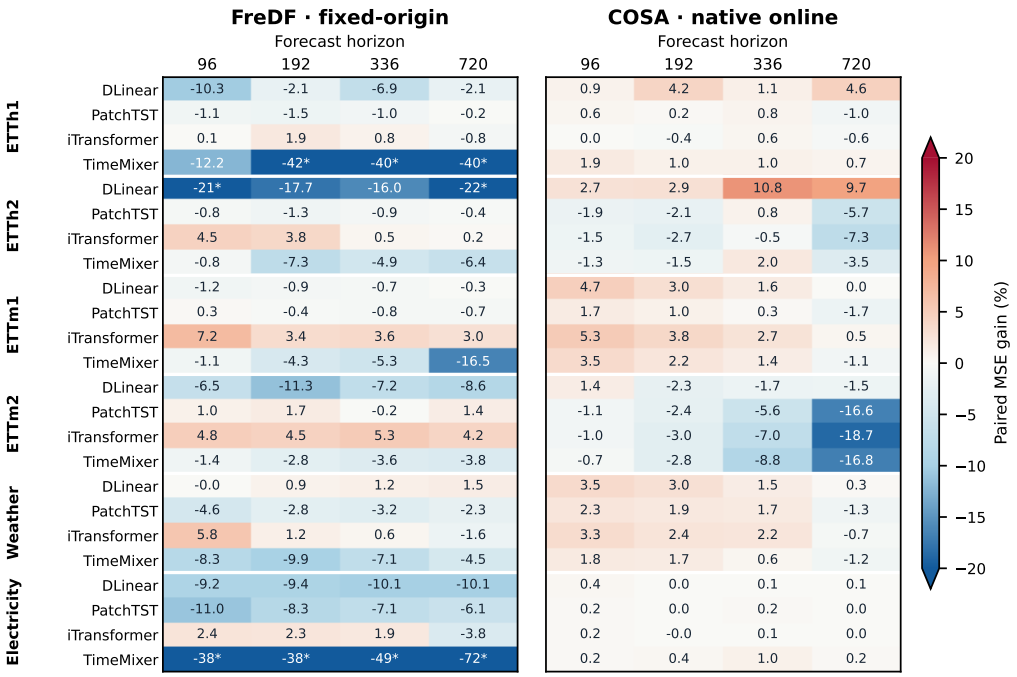

Figure 6: Three-run $L = 96$ paired MSE gains for forecast-once FreDF and native online-revision COSA. They are displayed together for coverage, not pooled into one forecasting contract.

B Implementation and Protocol Details

Method names alone do not identify the code path or execution contract evaluated in the benchmark. We therefore record source commits, licenses, integration variants, and the evaluation objects used by related benchmarks before listing recipe details.

Table 7: Implementation provenance for the primary $L = 96$ integration snapshots. GLAFF uses the paper-snapshot source here; the separate $L = 720$ and Traffic tracks use release-CLI commit `4dedf10e00`. Full source hashes and resolved variants are recorded with each run.

Method	Repository	Commit
GLAFF	`https://github.com/ForestsKing/GLAFF`	c91679403b
PIR	`https://github.com/icantnamemyself/PIR`	fc372bb020
GTR	`https://github.com/macovaseas/GTR`	d941619061
COSA	`https://github.com/bigbases/COSA_ICLR2026`	527c0feb9e
KUP-BI	`https://github.com/yihannnnn/KUP-BI`	da20530642
Delta-Adapter	`https://github.com/Anoise/Adapter`	0add06ea7b
LatentTSF	`https://github.com/Muyiiiii/LatentTSF`	7c8ae947ee
FAN	`https://github.com/wayne155/FAN`	838e1b002a
FreDF	`https://github.com/Master-PLC/FreDF`	ff7c571b24
TAFAS	`https://github.com/kimanki/TAFAS`	139bf98067

Table 8: Primary evaluation objects and questions in closely related forecasting benchmarks. The distinction concerns scope rather than superiority; normalization and other static transformations can fall within multiple frameworks.

Framework	Primary evaluated object	Treatment of context	Primary output
TSPP	End-to-end forecasting pipeline	Exposes data, feature, training, and model choices	Reproducible model/pipeline comparison
TimeRecipe	Modules in a canonical supervised architecture	Datasets, horizons, and task settings	Module effectiveness and architecture recipe
CombinationTS	Five composable architecture stages	Probabilistic evaluation-condition space	Marginalized performance/stability attribution
PlugBench-TS	Enhancement execution contract, including loss, retrieval, correction, and online state	Paired controlled context plus separate targeted author-setting checks	Claim–evidence record, including incompatibility and scoped costs

The records distinguish the evaluated integration from the method family. In particular, the δ-Adapter results refer to the frozen Ada-Y output component, not the complete input-and-output method. LatentTSF uses the selected MLP autoencoder/latent-predictor integration. GTR and KUP-BI include explicitly adapted backbone paths. These labels must accompany comparisons rather than being hidden under a family name.

Recipe and extension scope. The main suite fixes the four backbone recipes across plugin comparisons and resolves plugin-owned settings from pinned sources. Backbone recipes are selected from their own official sources before plugin outcomes are inspected; the controlled matrix does not import a different plugin-paper-specific backbone tuning into every comparison. Optimizers, schedulers, regularization, augmentation, and initialization can affect forecasting performance, but they are controlled rather than ranked here. This produces both source-scope recipes and explicit extensions. FAN has 24 author-reported DLinear cells and 72 backbone extensions; GTR has 12 source-scope Electricity cells and 84 extensions. FreDF has 64 source-scope and 32 extension cells. KUP-BI and LatentTSF are entirely extensions at controlled $L = 96$. GLAFF uses its pinned paper-snapshot plugin architecture; Ada-Y uses the pinned output-component recipe. The KUP-BI source uses $L = 336$ for DLinear/PatchTST and precomputes test-window correlations; our port keeps $L = 96$ and retrieves at forecast time so that retrieval work is charged to the forecast instead of hidden in an ahead-of-time test cache. Full output equivalence to the author's cached runner has not been established. Recipe classification is retained per cell in the released CSV.

Normalization-sensitive integrations. The FAN paper reports DLinear among its primary backbones and, in a separate TimesNet/Koopa study, removes built-in RevIN or replaces it with FAN. FAN is therefore treated as a normalization intervention, not as evidence that arbitrary normalization stacks are source-supported. Our PatchTST, iTransformer, and TimeMixer extensions retain the pinned backbone behavior: PatchTST uses RevIN, iTransformer performs built-in instance standardization, and TimeMixer uses its configured normalization layer. These cells test a registered direct-stacking extension at a fixed-backbone operating point; they are not source-setting reproductions. The replacement controls below separately retrain PatchTST with `revin=false`, TimeMixer with `use_norm=0`, and iTransformer with its built-in instance standardization disabled in an isolated code copy. The iTransformer switch leaves the normalization-on path bitwise identical to the pinned implementation; its normalization-off arm is a controlled extension, not an author-native recipe. We reserve source-setting numerical claims for separate author-native anchors. The controlled GTR–PatchTST integration likewise retains the benchmark's PatchTST configuration and RevIN, whereas the released GTR Electricity configuration disables RevIN. The final $L = 96$ GLAFF matrix uses the paper-snapshot plugin settings with controlled preprocessing; the author-setting study below examines the data-protocol difference, and the $L = 720$ track retains its separate release-CLI version.

Table 9: Three-arm FAN normalization control on ETTh2 over seeds 2021–2023. Entries are mean MSE gain ± sample standard deviation. "Stacked" compares canonical backbone+FAN with the canonical backbone; "increment" compares FAN replacement with a separately trained no-normalization scaffold; "net replacement" compares that replacement system with the canonical backbone. The latter two share the replacement system but use different bare references. This control retains our split, scaler, and backbone family and is not a source-table reproduction.

Backbone	H	Stacked vs. canonical	FAN increment	Net replacement
patchtst	96	-3.88 ± 2.54	-0.25 ± 2.38	-4.35 ± 2.85
patchtst	720	-64.52 ± 3.13	$+6.69 \pm 2.29$	-66.73 ± 1.70
timemixer	96	-2.14 ± 1.25	$+5.42 \pm 3.25$	-3.50 ± 3.86
timemixer	720	-61.78 ± 4.34	$+4.81 \pm 3.89$	-62.08 ± 3.24

The three-arm control prevents two different baselines from being conflated. At $H = 720$, FAN improves the separately trained no-normalization scaffolds by $+6.69\%$ and $+4.81\%$ for PatchTST and TimeMixer. Removing normalization, however, makes those scaffolds substantially worse than their canonical backbones, so the complete replacement systems remain -66.73% and -62.08% below the canonical references. Thus the control establishes normalization sensitivity in these unreported backbone extensions but does not show that removal alone recovers a competitive system. The separate author-runner anchors below verify that the released FAN path can be beneficial in its own protocol; neither set of rows replaces the controlled matrix. To check whether these four ETTh2 contexts were unusually selected, Table 10 extends the same replacement comparison to all six datasets and four horizons on PatchTST and TimeMixer. FAN improves the separately trained no-normalization scaffold in 17/24 PatchTST cells, but the complete replacement exceeds the canonical PatchTST in only 6/24. The corresponding TimeMixer counts are 10/24 and 3/24. These are single-seed descriptive comparisons; the ETTh2 control above supplies the independent-seed check. Per-cell values and configuration differences are released with the experiment record.

Table 10: FAN normalization-replacement extension at $L = 96$ (seed 2021; six datasets and four horizons per backbone). Entries give positive cells/24 and median MSE gain (%). "Increment" uses a separately trained no-normalization scaffold; "net" and "stack" use the canonical bare backbone. These pairings are controlled extensions.

Backbone	Increment over no norm	Net over canonical	Stack over canonical
PatchTST	$17/24, +1.98$	$6/24, -4.27$	$6/24, -4.12$
TimeMixer	$10/24, -1.98$	$3/24, -4.50$	$3/24, -4.05$

The iTransformer control asks the same replacement question for its built-in standardization. Table 11 reports both FAN's increment over the separately trained no-normalization backbone and the resulting system's gain over the original normalized backbone. Across 24 three-seed contexts, FAN improves the no-normalization backbone in 19, but the complete replacement exceeds the canonical normalized backbone in 10; direct stacking does so in 9. This comparison does not make either integration uniformly preferable. Per-seed MSEs are in `data/fan_itransformer_replacement_l96_per_seed.csv`; resolved configurations remain in the experiment record.

Table 11: FAN as a replacement for iTransformer instance standardization at $L = 96$. Each entry is the MSE gain (%) of no-norm+FAN over a separately trained no-norm bare model / over the canonical normalized bare model. Each gain is computed from three-seed mean MSEs. Positive values mean lower MSE. This is a controlled extension, not an author-native pairing.

Dataset	$H = 96$	$H = 192$	$H = 336$	$H = 720$
ETTh1	$+3.5/-4.1$	$+8.0/-3.0$	$+6.7/-5.6$	$+17.9/-11.4$
ETTh2	$+50.9/+0.6$	$+56.0/-1.5$	$+55.4/-15.3$	$+49.7/-69.0$
ETTm1	$+12.5/+4.7$	$+10.8/+2.1$	$+14.0/+0.3$	$+13.4/-0.9$
ETTm2	$+24.6/+3.8$	$+33.8/+4.4$	$+47.8/-3.8$	$+51.2/-26.5$
Weather	$-1.9/+3.3$	$+0.4/+3.6$	$+0.8/+3.6$	$+3.8/+6.7$
Electricity	$-23.6/-24.9$	$-17.7/-19.2$	$-10.7/-20.2$	$-12.9/-16.7$

Table 12: Targeted author-setting anchors, reported separately from the controlled matrix. FAN follows its released native 7/2/1 split and full-series scaling; the repository documents that this scaling uses future statistics. Mean and sample standard deviation are over three independently trained seeds. GTR uses three seeds of the released Electricity–PatchTST recipe with RevIN disabled (`revin=0`), `affine=0`, and cycle length 168 for both bare and enhanced models. These runs check the complete source setting, not the isolated effect of RevIN, and do not replace controlled cells.

Method	Dataset/backbone	H	Seeds	Bare MSE	Enhanced MSE	Gain (%)
GTR	Electricity/PatchTST	96	2021/2022/2023	0.1638±0.0004	0.1336±0.0002	+18.40±0.32
FAN	ETTh1/TimesNet	96	42/43/44	0.4954±0.0495	0.3637±0.0015	+26.13±6.93
FAN	ETTh2/DLinear	720	42/43/44	0.1747±0.0000	0.1590±0.0016	+9.03±0.93

Three further author-runner checks use the authors' released settings and our shared ETTh1 data file. KUP-BI slightly worsens this short-horizon DLinear result, as in its source table; FreDF and LatentTSF improve their matched controls. The single-seed rows test whether the released numerical paths behave as expected at these settings. They do not replace controlled $L = 96$ cells or establish output equivalence between the source runners and our ports.

Table 13: Additional ETTh1 author-runner checks. MSEs in the source columns are the corresponding paper values; our columns use one independently trained seed on the shared data file. KUP-BI's control disables auxiliary fusion within the released runner.

Method	Backbone	L/H	Paper control	Paper plugin	Our control	Our plugin
KUP-BI (Liu et al., 2026)	DLinear	336/96	0.384	0.386	0.3841	0.3858
FreDF (Wang et al., 2025)	iTransformer	96/96	0.391	0.377	0.3922	0.3776
LatentTSF (Yang et al., 2026)	DLinear	720/96	0.375	0.366	0.3754	0.3640

Table 14: Plugin recipes for the four-backbone $L = 96$ matrix. "Reported pairing" identifies source-reported dataset–backbone/lookback scope, not a full author-setting reproduction. Counts cover 96 contexts per method. Final resolved options and provenance are indexed in `docs/RESULT_MANIFEST.json`; the earlier recipe-audit CSV predates the GLAFF paper-snapshot rerun.

Method	Lifecycle	Recipe / extension scope	Plugin-owned configuration
GLAFF	Joint	96 controlled paper-snapshot cells, including unreported backbone pairings	Paper-snapshot plugin at commit `c916794`: $q = .75$, $d = 512$, $d_{ff} = 2048$, 8 heads, 2 layers, dropout .1; source joint training and validation losses. Data splits, train-only scaling, and backbone recipes follow the controlled suite. The release-CLI and current-HEAD versions are compared separately in Appendix F.

Method	Lifecycle	Recipe / extension scope	Plugin-owned configuration
PIR	Retrieval + joint refinement	67 reported pairing; 29 extension	Dataset/backbone/horizon lookup: refine width $\{32, 64, 128, 256, 512\}$, one layer, 4 heads, $k \in \{10, 50\}$, stride $\{1, 2\}$; 10 epochs, batch 32, lr $\{2, 5, 10, 20\} \times 10^{-5}$, patience 3 or 10. Missing pairs transfer a preregistered source-backbone recipe.
GTR	Joint module	12 reported pairing; 84 extension	Cycle ETTh$=$ 24, ETTm$=$ 96, Weather$=$ 144, Electricity$=$ 168; 30 epochs with released type-3 schedule. Electricity DLinear/PatchTST/iTransformer are reported pairings; other pairings transfer the dataset cycle and integration before test.
COSA	Online adaptation	96 online-revision cells; no emit-once cells	PAAS, partial-label adaptation, revised emission; 3 update steps, stream batch 48, context size 10. Reported only in the native online revision-output cohort.
KUP-BI	Retrieval + fusion	0 reported pairing; 96 extension	Dataset/horizon lookup for $\alpha \in [.35, .95]$, $\tau \in \{.01, .05, .15, 5, 10\}$ and top-$m \in \{1, 3, 7, 9\}$; chunk 128. Author DLinear/PatchTST scripts use $L = 336$, whereas this matrix fixes $L = 96$; unreported backbone ports are also labeled extensions.
δ-Adapter	Static output correction	96 controlled Ada-Y component cells, including unreported backbone pairings	Output-only Ada-Y component, width 512 and $\delta = .1$; 10 epochs, lr 10^{-4}, batch 16, patience 3, half-each-epoch schedule. This is not the full multiplicative X+Y runner.
LatentTSF	Latent training system	0 reported pairing; 96 controlled-lookback extension	Author dataset lookup gives latent width $\{32, 64, 512\}$ and AE hidden width $\{64, 128, 1024\}$; MSE/cosine weights $10/15$; AE pretraining 500 epochs and forecasting 100 epochs. The suite seed controls the paired context and model construction; the released stage seed 42 controls AE and forecasting optimization/data order. The author main recipe uses $L = 720$; the matrix fixes $L = 96$.
FAN	Joint normalization	24 reported pairing; 72 extension	Dataset-specific frequency K: ETTh1/2, ETTm1/2, Weather, and Electricity use $4/3$, $11/5$, 2, and 3, respectively; rFFT; 100 epochs, lr 3×10^{-4}, batch 32, patience 5, cosine schedule, weight decay 5×10^{-4}. DLinear is the reported pairing. The three extension backbones retain their pinned internal normalization, so these cells test direct stacking rather than the paper's TimesNet/Koopa replacement setting.
FreDF	Training loss	64 reported pairing; 32 extension	rFFT loss with dataset/backbone lookup for reconstruction weight $\{0, .1, .2, .4, 1\}$ and complementary auxiliary weight; epochs $\{3, 10\}$, lr $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$. Unreported pairs and source sweeps without a unique primary value use a preregistered release default.
TAFAS	Online adaptation	96 online-revision cells; no emit-once cells	PAAS, partial-label forecast revision; one update step, batch 64, hidden width 128, gate initialization .01, Adam lr .005, weight decay 10^{-4}. Reported only in the online-revision cohort.

Table 16: Executable contract-gate evidence. Each invalid mutation is expected to leave a valid control unchanged or to terminate before scoring. The online row is a synthetic lifecycle smoke trace rather than an accuracy result.

Gate	Invalid mutation or trace	Observed outcome
Data isolation	Change validation/test suffix values	Training scaler and inputs unchanged
Configuration	Add unknown option or reorder input rows	Unknown option rejected; row order retained
Artifact identity	Change resume config or tamper saved file	Both rejected; exact reload passes
Lifecycle	Send undeclared joint method to generic runner	Rejected before evaluation
Retrieval causality	Change observations after training-library endpoint	KUP-BI library and ranking unchanged
Online release	COSA, $H = 24$; three release events	3/3 allowed; latest requested labels precede release boundary

Table 15: Three-seed paired gains by plugin-recipe scope at $L = 96$. Source-scope identifies a source-reported pairing and plugin recipe under our controlled data and backbone settings; it is not an author-native numerical reproduction. The strata contain different contexts, so their difference is descriptive.

	Source-scope			Extension		
Method	n	Gain > 0	Median (%)	n	Gain > 0	Median (%)
PIR	67	59/67	+1.62	29	21/29	+3.97
GTR	12	10/12	+6.14	84	27/84	-2.47
FAN	24	15/24	+3.28	72	16/72	-3.95
FreDF	64	25/64	-0.74	32	3/32	-7.24

Bare-backbone options and optimization are resolved independently of plugin options. The initial seed-2021 backbone and plugin audits are released in `data/official4_l96_backbone_recipe_audit.csv` and `data/official4_l96_plugin_recipe_audit.csv`. For the final three-run matrix, the supplementary archive provides locked suite/shard configurations and a per-seed result ledger with configuration and source-hash signatures. The GLAFF paper-snapshot version supersedes its initial release-CLI row. In a joint lifecycle, the plugin-stage optimizer acts on all trainable enhanced-system parameters, including the backbone unless explicitly frozen; in a frozen lifecycle, the selected bare state and buffers must remain unchanged.

Adding a plugin. A new integration registers its resolved configuration, source provenance, capabilities, and lifecycle hooks. It then passes checks for finite outputs, matched targets and forecast timestamps, checkpoint reload, frozen-state preservation where applicable, and causal access to memory or released labels. Model-specific feature injection requires an explicit adapter. Accepted runs export the resolved configuration, selected checkpoint, predictions, timestamps, metrics, and scoped cost records so that the corresponding tables and figures can be regenerated without retraining. These checks establish consistency of the evaluated integration; they do not by themselves reproduce a source paper.

Executable gate checks. We exercise these rules with targeted implementation tests. Changing validation/test suffix values must leave train-fitted scaling and training inputs unchanged; row order is preserved and unknown configuration keys are rejected; exact checkpoint reload is required while changed resume configurations and tampered artifacts are rejected; an undeclared joint lifecycle cannot enter the generic runner; and changing post-training observations cannot change KUP-BI's training library or reference retrieval ranking. All five checks pass in the recorded CPU-only gate audit (Table 16). A separate COSA smoke run records three allowed release events and retains two unreleased batches at stream termination; targets are fetched only after the corresponding group deadline. These tests cover the named invariants and demonstrate active rejection, but the synthetic trace is not evidence of forecasting quality.

Third-party onboarding example. FreDF supplies a compact accepted example: the integration pins the released source, declares a training-loss lifecycle with no deployment state, implements only the registered loss hook, rejects unsupported options before fitting, and exports the same configuration, provenance, prediction, and cost records as other cells. All 96 controlled FreDF cells pass

through this path; author-reported and extension pairings remain labeled separately. Conversely, an intake check rejects PULSE as a plugin because its release is a complete forecasting model with its own backbone rather than a detachable enhancement. The supplementary README gives the registration and evaluation steps; admission concerns interface and evidence validity, not whether test accuracy is favorable.

Data and statistical units. Dataset hashes, ordered variables, train-only scaling parameters, and train/validation/test origin hashes are exported per run. Non-ETT splits are chronological; no duplicate timestamp is silently dropped when doing so would shift origins relative to an existing paired experiment. A seed repeat requires re-fitting the relevant bare and enhanced systems, not merely resampling test windows. Block resampling can assess uncertainty conditional on trained models, but cannot replace training-seed replication.

Admission and information access. A frozen method must preserve both parameters and relevant state, including normalization buffers. A joint method may change its backbone but must report the independent bare comparison as a system effect. Retrieval uses eligible past or training material under its declared policy. Online predictions are scored under an explicit label-release and forecast-revision convention. TAFAS results produced after online revision are reported separately from forecasts emitted once at each test timestamp. No invalid disable or checkpoint-replacement operation is scored as zero gain.

Evaluation-time event order. At each test forecast timestamp t, the input history is available before any target in $Y_{t+1:t+H}$. Methods in the non-revised cohort emit once and cannot use test labels for that forecast. Retrieval methods query only their declared training memory or causally available past observations before emission. In an online revision stream, the initial forecast is cached; a target becomes available for updating only at its recorded release time, after which the method may update state and revise an earlier forecast if its protocol permits. The run record identifies whether the initial or revised forecast is scored. Revised forecasts are therefore summarized separately from forecasts emitted once, even when their output tensors have the same shape.

C Formal Estimands and Measurement Limits

The following propositions are applications of conditional-expectation projection and quadratic optimization. They motivate controls and distinguish observable quantities from ideal limits; we do not claim a general performance guarantee for forecasting plugins. Unless stated otherwise, fix the data distribution, forecast horizon, scaling, and evaluation protocol. Forecast vectors have finite second moments. For forecast tensors, $\|z\|^2$ denotes $\|z\|_F^2/(HC)$ and the corresponding inner product has the same normalization; parameter and retrieval-score vectors use ordinary Euclidean norms. Predictions and learned parameters are treated as fixed when defining evaluation risk; fitting must not use evaluation labels. For online systems the risk is over a specified stream schedule and its causally available state.

C.1 Payoff and predictable residuals

Proposition 1 (interface-limited correction). Let $e = Y - \hat{Y}_B$, let Z be deployable information, and let $a(Z)$ be square-integrable. With $m(Z) = \mathbb{E}[e \mid Z]$,

$$\mathbb{E}\|e - a(Z)\|^2 = \mathbb{E}\|e - m(Z)\|^2 + \mathbb{E}\|m(Z) - a(Z)\|^2. \tag{6}$$

Consequently, Equation (1) holds, and the largest population improvement among all such corrections is $\mathbb{E}\|m(Z)\|^2$.

Proof. Expand $e - a = (e - m) + (m - a)$. The cross term vanishes because $\mathbb{E}[e - m \mid Z] = 0$. Set $a = 0$ and subtract the two decompositions. □

For an actual plugin correction δ, including a joint-system output difference, the directly measurable identity is

$$\mathbb{E}\|e\|^2 - \mathbb{E}\|e - \delta\|^2 = 2\mathbb{E}\langle e, \delta\rangle - \mathbb{E}\|\delta\|^2. \tag{7}$$

With nonzero norms, define $\rho = \mathbb{E}\langle e, \delta\rangle / (\sqrt{\mathbb{E}\|e\|^2}\sqrt{\mathbb{E}\|\delta\|^2})$ and $q = \sqrt{\mathbb{E}\|\delta\|^2/\mathbb{E}\|e\|^2}$. Relative improvement is $2\rho q - q^2$. Positive alignment alone is insufficient: improvement requires $\rho > 0$ and $0 < q < 2\rho$. These are descriptive population or empirical identities, not label-free deployment rules. Degenerate zero-norm cases are reported separately.

A validation probe estimates the performance of a specified fitted correction, not m or a certified upper bound. Train-only fitting and validation-only selection are required. In-sample backbone predictions can make residual probes optimistic; separate calibration windows or chronologically valid out-of-sample predictions should be examined before interpreting train–deployment predictability. Temporal dependence precludes treating overlapping origins as independent. Uncertainty from time-block resampling and variability across training seeds answer different questions.

C.2 Checkpoint-pairing dependence and local optimization coupling

Equation (2) follows by adding and subtracting the four risks. It does not require additivity of the underlying mechanisms. Its interpretation requires paired data and valid interventions: neutralization must preserve a meaningful computational path, and transplantation must document normalization, dimensions, persistent state, and downstream weights. A training-only loss intervention need not admit this experiment.

Proposition 2 (coupling in a quadratic model). Suppose a backbone parameter vector θ_0 is stationary with a neutral plugin, the risk is twice differentiable locally, and its quadratic surrogate is

$$Q(u, v) = R_0 + g^\top v + \tfrac{1}{2}u^\top A u + u^\top K v + \tfrac{1}{2}v^\top D v, \tag{8}$$

where u and v are backbone and plugin displacements. Assume the block Hessian $\left[\begin{smallmatrix} A & K \\ K^\top & D \end{smallmatrix}\right]$ is positive definite. Then $S = D - K^\top A^{-1} K$ is positive definite, and the optimal decreases in this surrogate are

$$\Delta_Q^{\text{frozen}} = \tfrac{1}{2} g^\top D^{-1} g, \qquad \Delta_Q^{\text{joint}} = \tfrac{1}{2} g^\top S^{-1} g \;\geq\; \Delta_Q^{\text{frozen}}. \tag{9}$$

For a fixed plugin displacement v, its best paired backbone displacement is $u^*(v) = -A^{-1}Kv$. Transplanting it back to $u = 0$ incurs surrogate loss

$$Q(0, v) - Q(u^*(v), v) = \tfrac{1}{2} v^\top K^\top A^{-1} K v \geq 0. \tag{10}$$

Proof. Complete the square in u:

$$Q(u, v) = R_0 + g^\top v + \tfrac{1}{2}v^\top S v + \tfrac{1}{2}(u + A^{-1}Kv)^\top A (u + A^{-1}Kv). \tag{11}$$

Minimizing first over u and then v gives the joint optimum; setting $u = 0$ gives the frozen optimum. Since $0 \prec S \preceq D$, $S^{-1} \succeq D^{-1}$. Substitution proves the transplant expression. □

This is an exact statement about Q, not about a globally nonconvex training trajectory. Applying it to the original risk requires small displacements and control of Taylor remainders. Neutral plugin initializations may be singular and actual Hessians indefinite; in these cases the assumptions need not hold. Adding damping changes the surrogate and must be disclosed. Existing final predictions do not estimate K or certify the approximation. Frozen-versus-joint fitting, local displacement experiments, and capacity-matched controls can test the proposed explanation; they do not establish it automatically. Transplant failure alone does not distinguish optimization coupling from an invalid interface.

C.3 Deployment frontiers and amortization accounting

Equation (3) defines an ideal feasible frontier for a specified implementation class, information set, and workload. The observed Pareto set includes only evaluated candidates. We compare risk and cost coordinates separately, allowing a predeclared risk tolerance when relevant; we do not assign subjective weights to form a universal ranking. Hardware, batch size, stream length, label-release delay, precision, and timing scope are part of the comparison. Unknown latency is not inferred from parameter count.

C.4 CONTRACT-PRESERVING TRANSFORMATIONS

The transformation protocol uses lifecycle equivalence rather than numerical equivalence:

Must remain unchanged	May change and must be measured
Admissible variables and availability times; memory-sample eligibility; label-release deadlines; update/batch/drain and emission schedule; functional role of state	Internal parameterization and rank; optimizer trajectory induced by that parameterization; pointwise predictions; measured risk and each resource coordinate

The same backbone checkpoint and trainable/frozen flags are part of the operating point. A transformation that removes an update or memory source is a contract-reducing substitution rather than a contract-preserving instance.

Fix an operating point ω and two predictors with the same output dimension. Let $d = P' - P$, $D(P', P) = \mathbb{E}\|d\|^2$, and $R(P) = \mathbb{E}\|P - Y\|^2$. Direct expansion yields

$$R(P') - R(P) = D(P', P) + 2\mathbb{E}\langle d, P - Y\rangle. \tag{12}$$

Cauchy–Schwarz then gives

$$|R(P') - R(P)| \le D(P', P) + 2\sqrt{R(P)D(P', P)}. \tag{13}$$

The inequality holds for expectations under the same distribution. Substituting an empirical validation distance into its right-hand side does not certify test risk without a generalization argument; validation selects a transformation and held-out test data measure its realized risk.

For a linear predictor Ax with rank-r truncated SVD A_r, Eckart–Young gives $\|A - A_r\|_2 = \sigma_{r+1}$. If $\mathbb{E}\|x\|^2 \le S$, then

$$D(A_r x, Ax) \le \sigma_{r+1}^2 S, \qquad |R(A_r x) - R(Ax)| \le \sigma_{r+1}^2 S + 2\sigma_{r+1}\sqrt{SR(Ax)}. \tag{14}$$

This controls only the transformed linear suboperator. Gates, normalization, optimization, and downstream nonlinearities require either additional Lipschitz bounds or paired measurement.

PIR uses cosine scores, and CPLM normalizes each key and query again after projection. Let unit-norm q, k_i have source score $s_i = q^\top k_i$. For orthogonal projection P_r, define $\alpha = \|P_r q\|_2$ and $\beta_i = \|P_r k_i\|_2$, both above the implementation's normalization epsilon. The projected cosine score is then $\tilde{s}_i = (P_r q)^\top (P_r k_i)/(\alpha\beta_i)$. Orthogonality and Cauchy–Schwarz give the conditional bound

$$|s_i - \tilde{s}_i| \le \sqrt{1 - \alpha^2}\sqrt{1 - \beta_i^2} + 1 - \alpha\beta_i =: \epsilon_i. \tag{15}$$

Indeed, the score difference is $q_\perp^\top k_{i,\perp} + (P_r q)^\top (P_r k_i)(1 - 1/(\alpha\beta_i))$. Below the normalization epsilon, only the trivial score-error bound of 2 is used. A query's top-k set is certified unchanged if every originally selected i and unselected j satisfy $s_i - s_j > \epsilon_i + \epsilon_j$. The study does not evaluate this margin certificate; it reports neighbor overlap and held-out prediction error instead. The bound is a geometric condition, not a forecasting payoff guarantee.

We apply the same verification obligations to online, static, and retrieval transformations. Table 17 gives bare, source-plugin, and reduced-plugin test MSE for every evaluated context, including those where the source plugin is harmful. Values average three seeds for display; the supplementary `data/cmpt_verification_cells.csv` retains each seed's MSE, selected rank, behavior check, and resource ratio. The original six CPLR contexts were selected after source-plugin test results existed, so their retained-gain results are descriptive. The registered $+1\%$ check compares reduced with source MSE; it does not require retaining 99% of source gain over bare.

Table 17: Absolute held-out test MSE for the same backbone checkpoint with no plugin (bare), the original plugin (source), and its reduced version. For CPLR, source is a separately replayed fixed-batch ($B = 48$) COSA reference, not the main-matrix PAAS run. Values are arithmetic means over seeds 2021, 2022, and 2023, including cells where the original plugin does not beat bare. The 90 individual seed values are in `data/cmpt_absolute_mse_cells.csv`; all comparisons use matched targets and origins.

Transformation	Dataset/H	Backbone	Bare	Source	Reduced
CPLR (studied)	ETTh2/$H$192	dlinear	0.43243	0.40102	0.41267
CPLR (studied)	ETTh2/$H$192	patchtst	0.37720	0.38040	0.37719
CPLR (studied)	ETTm1/$H$336	dlinear	0.41465	0.40883	0.40492
CPLR (studied)	ETTm1/$H$336	patchtst	0.39086	0.39071	0.38865
CPLR (studied)	Weather/$H$96	dlinear	0.19704	0.19001	0.19013
CPLR (studied)	Weather/$H$96	patchtst	0.17536	0.17127	0.17170
CPLR (new)	ETTh1/$H$96	dlinear	0.38686	0.38204	0.38485
CPLR (new)	ETTm2/$H$192	patchtst	0.24540	0.24810	0.24541
CPLR (arch.)	ETTh1/$H$192	itransformer	0.43464	0.43311	0.43464
CPLR (arch.)	ETTm2/$H$96	timemixer	0.17645	0.17650	0.17591
Ada-Y	ETTh2/$H$192	dlinear	0.43243	0.45240	0.43715
Ada-Y	ETTh2/$H$192	patchtst	0.37720	0.38490	0.38193
Ada-Y	ETTm1/$H$336	dlinear	0.41465	0.42172	0.42136
Ada-Y	ETTm1/$H$336	patchtst	0.39086	0.38959	0.39455
Ada-Y	Electricity/$H$96	dlinear	0.19460	0.18289	0.18923
Ada-Y	Electricity/$H$96	patchtst	0.16646	0.16024	0.16419
Ada-Y	Weather/$H$96	dlinear	0.19704	0.18237	0.18591
Ada-Y	Weather/$H$96	patchtst	0.17536	0.16564	0.16787
CPLM	ETTh2/$H$96	dlinear	0.31621	0.33619	0.33693
CPLM	ETTh2/$H$192	dlinear	0.43243	0.44464	0.44299
CPLM	ETTh2/$H$96	patchtst	0.29356	0.28435	0.28575
CPLM	ETTh2/$H$192	patchtst	0.37720	0.36600	0.36957
CPLM	Electricity/$H$96	dlinear	0.19460	0.17037	0.17633
CPLM	Electricity/$H$192	dlinear	0.19391	0.17542	0.17936
CPLM	Electricity/$H$96	patchtst	0.16646	0.16385	0.16874
CPLM	Electricity/$H$192	patchtst	0.17473	0.17219	0.17572
CPLM	Weather/$H$96	dlinear	0.19704	0.17846	0.17874
CPLM	Weather/$H$192	dlinear	0.23792	0.21595	0.21761
CPLM	Weather/$H$96	patchtst	0.17536	0.16749	0.16836
CPLM	Weather/$H$192	patchtst	0.22209	0.21461	0.21533

KUP-BI index layout. As a second retrieval-state instance, V2 caches the query layout of our on-demand KUP-BI port while retaining its original tie-resolution view. The train-only library, retrieval rule, and frozen model remain unchanged. In 12 fixed contexts, full test-loader replay gives bitwise-identical original and V2 forecasts; their MSE and paired gains are therefore identical. Persistent host-index arrays are 1.328× smaller, and the median V2/original full-system validation-batch latency ratio is 0.274 on an RTX 3090. Three source integrations gain at least 1% over bare. Per-context MSE and cost ratios are in `data/kup_v2_cmpt_cells.csv`. This is a transformation of our on-demand port, not the author's precomputed-correlation runner; Figure 3 and Table 4 retain the original port's costs. Low-rank Ada-Y uses ranks $\{2, 4, 8, 16\}$, CPLR uses $\{1, 2, 4, 8, 16\}$, and CPLM uses $\{8, 16, 32, 64\}$; each rank is selected on validation before the candidate's held-out test evaluation. CPLM keeps the trained PIR checkpoint fixed and covers 12 contexts spanning three datasets, two backbones, and two horizons at three seeds. The follow-up retains the original seed-2021 Electricity failures and does not change ranks after observing test outcomes.

The Table 5 grids differ because each plugin type was tested in valid, type-specific contexts. They are case studies, not a matched method ranking. For the 16 new Ada-Y cells, the lifecycle/behavior check passes in 16/16. A separate validation forecast-error check finds a rank within $+1\%$ of the dense source in 8/16; when none qualifies, the preregistered best-validation-rank fallback is still evaluated and flagged. Across all three seeds, only 9/24 Ada-Y cells meet the $+1\%$ source-relative test-MSE margin. Validation behavior, validation forecast error, and held-out forecast error are distinct results.

The additional CPLR follow-up uses ETTh1/DLinear/$H = 96$ and ETTm2/PatchTST/$H = 192$, three seeds each, outside the six previously studied contexts. It passes the test-MSE margin and causal event/frozen-backbone controls in 6/6 cells. The source plugin improves on bare by at least 1% in only two cells. A separate iTransformer/TimeMixer check adds six seed cells; all meet the source-relative MSE margin, but none has a source gain above 1%. These checks extend the implementation evidence without adding new CMPT transformations or establishing retained positive payoff on those two backbones.

For CPLM, exact top-k set agreement is measured on the first 32 validation histories per cell. Its median is 6.0% among the 17 useful-source cells that pass the source-relative test-MSE margin. The contract retains key eligibility and query timing, but does not require the same neighbors. Held-out MSE measures whether the changed retrieval preserves useful forecasts. All 12 Electricity cells miss the registered margin.

On Electricity/DLinear/$H = 720$, the recorded optimizer state is 1.352 GB for full COSA and 7.43 MB for CPLR under the same optimizer and precision. This is a state-size comparison, not an end-to-end timing result. Tables retain harmful source-plugin cells, for which retained payoff is undefined.

The online contract is checked on a real Weather/DLinear/$L = 96, H = 96$ stream rather than inferred from the implementation. We perturb only targets whose timestamps fall after a locked cutoff. Both full COSA and CPLR preserve all 5,280 forecasts emitted before that cutoff bitwise, retain the same release schedule, and change forecasts after the perturbed labels become eligible. This verifies the claimed causal boundary for that execution trace; it does not establish a forecasting gain by itself.

C.5 Evidence requirements and non-identifiable quantities

Evaluation question	Required evidence	Interpretation limit
Execution validity	Resolved contract, provenance, event trace, targets, and origins	Validity of a declared integration is not source-paper reproduction.
Payoff	Paired truth, origins, bare/plugin forecasts; seed and protocol records	Conditional-mean risk is not measured by a fitted probe.
Checkpoint pairing	Valid full, disabled, replaced-checkpoint, and bare evaluations where admissible	Four risks do not identify a Hessian or causal allocation.
Burden	Scoped timing, storage, memory, and offline logs	Evaluated Pareto set is not a universal optimum.

Analyses should distinguish direct calculations, measured interventions, fitted diagnostics, and unavailable quantities. Missing controls remain unassessed. Historical cohorts with different backbone configurations, information schedules, or data protocols are not silently pooled. New mechanistic hypotheses require preregistered evaluation beyond the development contexts. None of these propositions establishes that a particular implementation satisfies its assumptions or that the benchmark software already computes every diagnostic.

D Additional Pairing Diagnostic

Before interpreting cross-checkpoint replacement, we verify that the production extraction and checkpoint path can reconstruct the original paired system. Table 18 covers GTR on PatchTST and DLinear plus PIR and FAN on DLinear. Self-transplant and serialized reload reproduce every tested prediction exactly; replacing the backbone with the independently trained paired bare checkpoint changes predictions in every negative control. This validates the tested assembly path, not the scientific portability of a fitted plugin.

Table 18: Production self-round-trip validity check. Each fitted plugin is reattached to its own paired backbone, serialized through the whole-system checkpoint path, and reloaded on the same device. Self-transplant and reload columns report maximum absolute prediction difference from the original system; ΔMSE is the maximum absolute MSE change across those two reconstructions. The final column reports maximum absolute prediction difference after replacing the backbone with the independently trained paired bare checkpoint. This validates extraction and assembly, not portability to the independent checkpoint.

Plugin	Backbone	Context (D, H)	Self	Reload	ΔMSE	Independent bare
GTR	PatchTST	Weather, 192	0	0	0	13.009
GTR	DLinear	ETTh2, 192	0	0	0	2.169
PIR	DLinear	Weather, 96	0	0	0	14.744
FAN	DLinear	ETTh2, 96	0	0	0	1.598

The main runs already export the paired-backbone-only prediction as `joint_only`, so A, B, C, and D are available for every admitted replacement. Table 19 reports the resulting interaction after normalization by bare MSE. Large positive values mean that the full paired system performs better than the sum of its backbone-only and transplanted improvements under these operations. They do not identify why: for an input-transforming plugin, bypassing the plugin also exposes the jointly trained backbone to a different input distribution.

Table 19: Observed four-system interaction in Equation (2), normalized by paired bare MSE and reported in percentage points. Each context averages three independent seeds before computing the contrast. The favorable subset has observed full-system test gain $\geq 1\%$; it is descriptive.

	All contexts			Favorable full systems		
Method	n	Median I	$I > 0$	n	Median I	$I > 0$
GLAFF	96	+2.06	94/96	13	+3.66	13/13
PIR	96	+5.96	90/96	64	+11.74	63/64
GTR	96	+39.59	94/96	35	+42.43	35/35
KUP-BI	96	+0.06	68/96	13	+0.75	9/13
FAN	96	+58.49	96/96	28	+96.89	28/28

We also exchanged fitted plugin states between three jointly trained runs of the same dataset, backbone, and horizon, holding each target checkpoint fixed when comparing its own state with another seed's state. Table 20 reports the median of the six ordered exchanges within each context, followed by the median across eight contexts per plugin. All 72 target-system replays reproduced their saved predictions exactly. FAN's error stayed close to its native pairing; PIR showed a modest median increase, whereas GTR sometimes deteriorated substantially. All eight GTR contexts are benchmark extensions, and only 2/48 GTR exchanges joined source and target runs that both had at least 1% observed test gain over bare. This follow-up does not establish behavior in source-reported GTR settings or identify the cause of replacement loss.

Table 20: Plugin-state exchange between jointly trained systems of the same architecture. Positive ΔMSE means that replacing the target's own plugin state increases its test error. Each method has eight dataset–backbone–horizon contexts, three seeds per context, and six ordered exchanges per context; ordered exchanges are not independent replications.

Plugin	Ordered exchanges	Median ΔMSE (%)	Contexts with ΔMSE > 0
GTR	48	+68.78	6/8
FAN	48	+0.03	7/8
PIR	48	+2.36	6/8

Aggregate transplant results can still hide whether checkpoint dependence is concentrated in a particular backbone. The following breakdown retains the same intervention and reports it by backbone.

Table 21: GTR same-architecture checkpoint replacement by backbone in the three-seed $L = 96$ matrix. The eligible subset has observed full-system test gain $\geq 1\%$; gain retention is descriptive and uses the paired bare reference.

Backbone	Full positive	Eligible	Replacement median	Retain $\geq 50\%$
DLinear	14/24	14	+3.31	3/14
PatchTST	8/24	7	-616.84	0/7
iTransformer	9/24	8	-1.15	1/8
TimeMixer	6/24	6	-112.43	0/6

Table 22 further separates GTR and FAN replacement outcomes by source-scope and extension pairings, without selecting on the observed test gain.

Table 22: Same-architecture checkpoint replacement for GTR and FAN by plugin-recipe scope. Context gains use the three matched seeds before medians are taken. GTR source-scope rows split Electricity by backbone. Source-scope is not an author-native run; no source is selected by test gain.

Pairing	Recipe scope	Contexts	Full med. (%)	Replaced med. (%)	Replaced > 0
GTR	Source-scope	12	+6.14	-10.76	4/12
DLinear	Source-scope	4	+16.94	+4.52	4/4
PatchTST	Source-scope	4	+3.85	-753.32	0/4
iTransformer	Source-scope	4	+3.23	-10.76	0/4
GTR	Extension	84	-2.47	-26.10	8/84
FAN	Source-scope	24	+3.28	-1.83	10/24
FAN	Extension	72	-3.95	-10.61	5/72

E Coverage and Complete Controlled Results

The leave-one-dataset-out diagnostic asks whether one dataset alone drives the aggregate observations. Across all six omissions, all eight forecast-once methods retain both positive and negative context gains; the median replacement gain remains negative, and the favorable-source replacement positive rate ranges from 29.6% to 38.9% (Table 23). This is a deterministic sensitivity analysis, not a sampling-based confidence interval.

Table 23: Leave-one-dataset-out sensitivity of the complete three-seed $L = 96$ matrix. Ranges span the six omitted datasets. The favorable replacement rate conditions on observed full-system test gain $\geq 1\%$ and is descriptive.

Diagnostic	Minimum	Maximum
Methods with mixed payoff	8/8	8/8
Replacement median gain (%)	-3.90	-2.46
Favorable replacement positive rate (%)	29.6	38.9

Methods that revise forecasts after labels are released cannot be pooled with forecasts emitted once at the test timestamp, so their aggregate results are reported separately before the complete controlled matrix.

Table 24: Three-seed native online-revision cohorts. Each context gain uses its own paired three-seed stream; these outcomes are separate from non-revised forecasts.

Method	Done/96	Positive	Median gain (%)
COSA	96/96	62	+0.24
TAFAS	96/96	43	-0.18

Table 25 lists every COSA and TAFAS context using its three-run paired gain under the respective online-revision protocol. Together with the four forecast-once backbone tables below, it completes

the ten-plugin $L = 96$ result listing. Per-run bare and enhanced MSEs are supplied in the accompanying result ledger.

Table 25: Three-seed paired MSE gain (%) for native online-revision integrations at $L = 96$. Each method retains its own chronological update and emission rule and is compared only with its paired bare forecast; the columns are not a direct method ranking.

Dataset	H	COSA gain	TAFAS gain
DLinear			
ETTh1	96	+0.93	+0.20
ETTh1	192	+4.18	+3.59
ETTh1	336	+1.12	+1.89
ETTh1	720	+4.65	+3.72
ETTh2	96	+2.68	-5.16
ETTh2	192	+2.87	-1.90
ETTh2	336	+10.81	+12.01
ETTh2	720	+9.72	+8.02
ETTm1	96	+4.69	+3.66
ETTm1	192	+3.02	+2.08
ETTm1	336	+1.59	+1.72
ETTm1	720	+0.02	+1.62
ETTm2	96	+1.43	-1.03
ETTm2	192	-2.29	-7.21
ETTm2	336	-1.66	-7.15
ETTm2	720	-1.47	-0.89
Weather	96	+3.52	+3.74
Weather	192	+2.95	+2.87
Weather	336	+1.50	+2.58
Weather	720	+0.26	+3.20
Electricity	96	+0.37	+1.10
Electricity	192	+0.01	+0.62
Electricity	336	+0.09	+0.79
Electricity	720	+0.13	+0.46
PatchTST			
ETTh1	96	+0.62	-0.15
ETTh1	192	+0.24	-0.05
ETTh1	336	+0.77	-1.64
ETTh1	720	-1.04	-0.39
ETTh2	96	-1.86	-3.29
ETTh2	192	-2.12	-5.02
ETTh2	336	+0.81	-5.57
ETTh2	720	-5.72	-2.89
ETTm1	96	+1.71	-0.38
ETTm1	192	+1.00	+0.13
ETTm1	336	+0.25	+0.06
ETTm1	720	-1.71	-1.04
ETTm2	96	-1.08	-1.42
ETTm2	192	-2.39	-2.76
ETTm2	336	-5.63	-1.23
ETTm2	720	-16.59	-10.26
Weather	96	+2.30	+0.62
Weather	192	+1.88	+0.77
Weather	336	+1.73	-1.08
Weather	720	-1.27	-8.86
Electricity	96	+0.19	+0.26
Electricity	192	+0.00	+0.11
Electricity	336	+0.16	-0.17
Electricity	720	+0.03	+0.11
iTransformer			
ETTh1	96	+0.03	-0.75
ETTh1	192	-0.41	-0.40
ETTh1	336	+0.55	+0.30
ETTh1	720	-0.59	-0.01
ETTh2	96	-1.46	-8.27
ETTh2	192	-2.70	-9.01
ETTh2	336	-0.52	-3.88
ETTh2	720	-7.30	-9.23
ETTm1	96	+5.27	+3.36
ETTm1	192	+3.75	+1.37
ETTm1	336	+2.66	+0.57
ETTm1	720	+0.52	-0.11
ETTm2	96	-1.04	-7.53
ETTm2	192	-2.96	-9.28
ETTm2	336	-7.03	-8.34
ETTm2	720	-18.74	-17.82
Weather	96	+3.31	+3.80
Weather	192	+2.44	+1.89
Weather	336	+2.25	+0.79

Dataset	H	COSA gain	TAFAS gain
Weather	720	-0.67	+0.59
Electricity	96	+0.18	+0.53
Electricity	192	-0.00	-0.21
Electricity	336	+0.12	-1.59
Electricity	720	+0.04	-0.66
TimeMixer			
ETTh1	96	+1.86	+0.88
ETTh1	192	+1.01	+0.27
ETTh1	336	+1.03	-0.39
ETTh1	720	+0.65	+1.13
ETTh2	96	-1.26	-8.06
ETTh2	192	-1.48	-9.32
ETTh2	336	+1.99	-0.20
ETTh2	720	-3.55	-3.04
ETTm1	96	+3.47	+1.87
ETTm1	192	+2.20	+1.03
ETTm1	336	+1.44	+0.50
ETTm1	720	-1.09	-0.47
ETTm2	96	-0.70	-4.96
ETTm2	192	-2.83	-14.34
ETTm2	336	-8.76	-14.49
ETTm2	720	-16.82	-43.44
Weather	96	+1.78	-1.13
Weather	192	+1.71	-0.78
Weather	336	+0.59	-2.33
Weather	720	-1.21	-6.28
Electricity	96	+0.24	+0.01
Electricity	192	+0.39	+0.24
Electricity	336	+0.96	+0.29
Electricity	720	+0.20	-2.46

The six-dataset $L = 96$ grid contains 768 forecast-once contexts and 192 native online-revision contexts, each with three accepted plugin runs and its corresponding paired bare runs. Traffic and Exchange are outside this primary grid. A plugin may use a different paired bare checkpoint even when dataset, backbone, horizon, and seed match another plugin; the tables therefore report within-plugin paired gains rather than imply one shared bare prediction for all methods.

Table 26: DLinear: three-seed paired MSE gain (%) at $L = 96$. Each plugin uses its own paired bare run; no common bare checkpoint or cross-plugin accuracy ranking is implied. Mean bare and plugin MSEs are retained in the accompanying context ledger.

Dataset	H	GLAFF	PIR	GTR	KUP-BI	Ada-Y	LatentTSF	FAN	FreDF
ETTh1	96	-13.27	+3.60	+1.10	-2.38	-3.71	+1.14	-3.13	-10.34
ETTh1	192	-7.76	+6.34	+9.79	+4.11	-3.28	+7.91	+3.86	-2.08
ETTh1	336	-14.88	+0.48	+5.13	-1.33	-4.80	+3.04	-4.36	-6.87
ETTh1	720	-8.71	-2.01	+9.25	-1.59	-4.38	+10.19	-3.96	-2.10
ETTh2	96	-12.57	-6.32	-5.22	+1.17	-1.67	+3.99	+4.91	-20.61
ETTh2	192	-7.05	-2.82	-6.06	+4.72	-4.62	+5.84	+14.89	-17.73
ETTh2	336	+1.44	-1.32	-6.09	+2.62	-5.26	+9.95	+11.19	-16.02
ETTh2	720	+19.30	+4.87	-10.70	+7.82	-1.91	+18.88	-0.61	-22.41
ETTm1	96	-9.07	+10.43	+5.34	+0.26	+1.94	+0.24	+3.99	-1.19
ETTm1	192	-13.17	+6.78	+6.01	+0.27	-0.49	+0.62	+1.94	-0.87
ETTm1	336	-18.76	+6.70	+6.05	+0.13	-1.67	+0.46	-0.35	-0.73
ETTm1	720	-11.76	+4.44	+8.31	-0.07	-1.94	+0.32	-2.87	-0.28
ETTm2	96	+3.32	-0.32	+4.21	-0.58	+1.33	+1.78	+5.29	-6.49
ETTm2	192	-0.22	-16.72	-4.40	-1.69	+0.49	-3.66	+3.41	-11.29
ETTm2	336	+2.49	-26.97	+7.35	+5.02	+0.61	+4.51	+3.60	-7.17
ETTm2	720	+11.85	-10.65	-1.20	+4.63	-4.92	+12.96	-3.67	-8.59
Weather	96	+3.93	+9.43	-6.35	-0.64	+7.52	-0.73	+10.97	-0.00
Weather	192	+1.62	+9.23	-6.96	-0.38	+6.30	+2.02	+7.90	+0.90
Weather	336	+1.97	+5.46	-6.51	+0.09	+5.06	+1.99	+4.77	+1.23
Weather	720	+1.72	+2.57	-6.49	-0.52	+3.80	+2.65	+3.16	+1.50
Electricity	96	+14.75	+12.45	+26.78	-15.33	+6.18	-4.33	+4.24	-9.22
Electricity	192	+0.83	+9.53	+18.25	-15.73	+4.49	-5.84	+0.64	-9.45
Electricity	336	-5.30	+9.26	+15.63	-14.85	+5.28	-5.96	-1.26	-10.09
Electricity	720	-6.70	+9.34	+14.98	-12.39	+4.73	-5.83	-2.30	-10.11

Table 27: PatchTST: three-seed paired MSE gain (%) at $L = 96$. Each plugin uses its own paired bare run; no common bare checkpoint or cross-plugin accuracy ranking is implied. Mean bare and plugin MSEs are retained in the accompanying context ledger.

Dataset	H	GLAFF	PIR	GTR	KUP-BI	Ada-Y	LatentTSF	FAN	FreDF
ETTh1	96	-11.25	+6.23	-15.46	-0.22	-0.51	+2.57	-1.56	-1.10
ETTh1	192	-7.96	+5.55	-10.59	-0.09	-0.01	+2.57	-1.30	-1.46
ETTh1	336	-9.11	+3.97	-33.04	-0.08	-1.20	+2.16	-4.01	-0.95
ETTh1	720	-19.32	+0.67	-16.68	-0.11	-1.03	-0.37	-13.91	-0.23
ETTh2	96	-28.14	+3.14	-74.27	-0.26	-0.87	+4.01	-3.88	-0.75
ETTh2	192	-19.15	+2.97	-50.13	+0.06	-2.02	+2.83	-1.81	-1.35
ETTh2	336	-12.90	+1.09	-98.62	-0.19	-1.31	+0.38	-14.37	-0.87
ETTh2	720	-23.88	+0.91	-96.33	-0.91	-3.63	+1.38	-64.52	-0.36
ETTm1	96	-3.56	+1.62	+4.45	-0.34	+2.33	+1.41	-3.25	+0.28
ETTm1	192	-4.94	+1.55	+0.05	-0.79	+0.73	+2.32	-3.68	-0.40
ETTm1	336	-9.86	+2.05	-2.13	-1.00	+0.30	+1.37	-6.35	-0.78
ETTm1	720	-14.87	+1.53	+5.30	-0.98	+0.34	+1.49	-7.40	-0.66
ETTm2	96	+2.24	+0.66	-8.68	-1.19	+0.51	+2.51	-0.83	+0.96
ETTm2	192	+0.27	-0.54	-42.29	+1.17	+0.61	+3.04	+1.96	+1.72
ETTm2	336	-3.25	-0.24	-32.97	-0.41	+0.42	+1.72	-8.45	-0.24
ETTm2	720	-5.49	-3.08	-184.87	+0.17	+0.16	+4.14	-29.48	+1.43
Weather	96	-5.59	+4.49	+2.31	+4.17	+5.55	-3.45	+0.44	-4.58
Weather	192	-5.42	+3.37	-1.90	+3.23	+5.21	-1.71	+1.55	-2.80
Weather	336	-6.67	+2.38	+5.39	+1.44	+4.18	-0.33	+2.45	-3.18
Weather	720	-4.04	-0.16	-2.21	+0.48	+4.54	+1.55	+4.54	-2.29
Electricity	96	+0.94	+1.56	+8.63	-3.58	+3.91	-9.80	-11.54	-11.04
Electricity	192	-9.19	+1.45	-3.72	-4.09	+4.05	-9.24	-10.57	-8.31
Electricity	336	-14.11	+2.99	+3.64	-3.52	+4.92	-8.67	-9.63	-7.09
Electricity	720	-11.81	+2.56	+4.06	-5.81	+5.80	-5.90	-6.72	-6.09

Table 28: iTransformer: three-seed paired MSE gain (%) at $L = 96$. Each plugin uses its own paired bare run; no common bare checkpoint or cross-plugin accuracy ranking is implied. Mean bare and plugin MSEs are retained in the accompanying context ledger.

Dataset	H	GLAFF	PIR	GTR	KUP-BI	Ada-Y	LatentTSF	FAN	FreDF
ETTh1	96	-3.06	+2.07	-1.89	-0.37	-0.87	+0.60	-5.25	+0.11
ETTh1	192	-1.43	+3.24	-0.17	+0.08	-0.50	+1.12	-3.18	+1.93
ETTh1	336	-0.80	+2.85	-2.24	-0.75	-1.55	+1.53	-5.67	+0.84
ETTh1	720	-0.28	+8.04	+1.98	+2.06	-0.79	+6.94	-11.61	-0.84
ETTh2	96	-2.71	+0.99	-2.75	-0.22	-0.64	+4.32	-2.41	+4.48
ETTh2	192	-10.03	+1.97	-0.29	+0.22	-1.50	+1.53	-4.40	+3.82
ETTh2	336	-13.12	+0.40	-0.49	-0.86	-1.34	-1.07	-15.08	+0.48
ETTh2	720	-16.75	+0.96	-1.99	-0.99	-2.50	+1.15	-67.59	+0.20
ETTm1	96	-2.19	+9.96	+8.37	+0.75	+2.17	+1.96	+4.73	+7.16
ETTm1	192	-5.70	+5.80	+3.51	+0.23	+1.35	+0.96	+2.12	+3.43
ETTm1	336	-14.30	+6.70	-0.82	-0.15	+1.37	+1.43	+0.20	+3.63
ETTm1	720	-15.05	+3.57	-0.27	+0.43	+1.69	+0.87	-0.96	+2.96
ETTm2	96	+2.81	+3.18	+1.99	+1.12	+0.68	+4.91	+3.79	+4.85
ETTm2	192	-0.10	+2.12	+0.37	-0.73	+0.69	+3.04	+4.42	+4.52
ETTm2	336	-0.92	+0.34	+1.34	+0.95	+0.77	+5.10	-4.04	+5.30
ETTm2	720	-4.92	-1.63	-0.36	+0.40	+0.40	+5.00	-26.10	+4.21
Weather	96	+2.96	+3.70	-10.88	+0.82	+5.50	-6.96	+2.67	+5.84
Weather	192	-0.63	+2.09	-10.53	-0.03	+4.63	-1.85	+3.04	+1.22
Weather	336	-3.60	+1.48	-7.45	+0.34	+4.05	+1.40	+3.36	+0.62
Weather	720	-2.69	+0.27	-2.65	+0.07	+3.61	-0.04	+6.14	-1.64
Electricity	96	-5.78	+2.58	+8.21	-10.29	+2.16	-8.48	-25.27	+2.42
Electricity	192	-13.08	+1.55	+3.95	-8.61	+2.27	-7.30	-19.53	+2.26
Electricity	336	-17.50	+1.45	+2.51	-9.67	+2.74	-4.78	-18.24	+1.94
Electricity	720	-20.84	+1.62	-2.12	-9.01	+3.81	-0.81	-16.92	-3.84

Table 29: TimeMixer: three-seed paired MSE gain (%) at $L = 96$. Each plugin uses its own paired bare run; no common bare checkpoint or cross-plugin accuracy ranking is implied. Mean bare and plugin MSEs are retained in the accompanying context ledger.

Dataset	H	GLAFF	PIR	GTR	KUP-BI	Ada-Y	LatentTSF	FAN	FreDF
ETTh1	96	-9.75	+5.51	+1.85	-5.22	+0.17	+2.45	-1.65	-12.24
ETTh1	192	-16.82	+5.53	-8.70	-7.67	-0.41	+3.93	-0.59	-42.20
ETTh1	336	-16.92	+3.94	-9.13	-3.85	-1.57	+5.81	-1.63	-39.80
ETTh1	720	-28.51	+4.75	-13.66	-3.05	-0.48	+8.87	-5.97	-40.39
ETTh2	96	-11.27	+2.53	-6.11	-0.39	-0.11	+3.70	-2.13	-0.82
ETTh2	192	-40.58	+3.09	-12.52	-1.78	-0.54	+4.66	+2.12	-7.34
ETTh2	336	-11.28	+1.56	-14.05	+0.44	-1.05	+7.40	-9.42	-4.86
ETTh2	720	-8.76	+0.61	-22.21	+0.35	-2.86	+5.33	-61.71	-6.40
ETTm1	96	-9.97	+3.43	+5.08	-2.70	+2.21	-2.67	-1.60	-1.12

Dataset	H	GLAFF	PIR	GTR	KUP-BI	Ada-Y	LatentTSF	FAN	FreDF
ETTm1	192	-14.09	+2.66	+1.89	-2.53	+0.79	+0.55	-2.90	-4.32
ETTm1	336	-25.43	+3.55	-0.27	-2.43	+0.62	+1.55	-4.59	-5.35
ETTm1	720	-28.91	+1.92	-3.98	-3.27	+0.55	+0.82	-6.45	-16.48
ETTm2	96	-0.02	+0.80	-5.97	-1.23	+0.16	+3.28	-1.74	-1.38
ETTm2	192	-3.53	+0.50	-5.64	-1.47	+1.12	+0.18	-0.51	-2.77
ETTm2	336	-7.70	-0.55	-160.81	-2.33	+1.25	+0.70	-11.14	-3.59
ETTm2	720	-16.14	-2.23	-83.29	-1.52	+1.40	+1.02	-34.69	-3.84
Weather	96	-6.52	+0.46	-5.92	-9.27	+3.79	-7.86	-4.80	-8.27
Weather	192	-9.76	+0.42	+2.05	-7.92	+3.75	-5.79	-2.38	-9.91
Weather	336	-14.39	+0.25	-3.41	-6.45	+3.81	-3.99	-1.33	-7.15
Weather	720	-8.31	-0.69	-3.58	-2.83	+4.15	-1.00	+2.67	-4.51
Electricity	96	-6.95	+1.53	+5.13	-36.60	+3.94	-13.43	-19.56	-37.93
Electricity	192	-13.30	+0.71	-2.28	-29.70	+3.85	-9.73	-13.03	-38.10
Electricity	336	-20.89	+1.27	-18.37	-30.02	+3.94	-6.40	-9.97	-48.82
Electricity	720	-21.34	+4.63	+1.36	-25.69	+5.77	-5.88	-7.48	-71.89

E.1 TRAFFIC EXTENSION

Traffic provides a separate dataset extension of the $L = 96$ evaluation. Table 30 gives every planned plugin context at seed 2021: 145 of 160 plugin cells complete, and 15 report CUDA out-of-memory on the locked RTX 3090 configuration. The eight methods that emit forecasts once and the two online-revision methods are shown in separate column groups. These results are not included in the six-dataset main-matrix summaries.

Table 30: Exploratory Traffic results at $L = 96$, seed 2021: paired test-MSE gain (%). The eight forecast-once integrations and two native online-revision methods have different emission protocols and are not pooled. O marks a verified RTX 3090 CUDA out-of-memory failure, not zero gain. Traffic is outside the six-dataset main matrix; GLAFF here uses its release-CLI version, not the paper-snapshot version used in the final main matrix.

		Forecast once								Online revision	
Backbone	H	GLAFF	PIR	GTR	KUP-BI	Ada-Y	LatentTSF	FAN	FreDF	COSA	TAFAS
DLinear	96	+21.56	+27.32	+27.53	-15.70	+4.98	-8.57	+20.07	-16.03	+0.64	+1.40
	192	+15.14	+19.21	+20.50	-17.46	+4.23	-9.65	+13.47	-17.33	+0.48	+0.93
	336	+14.53	+17.82	+18.54	-17.68	+3.77	-10.36	+12.61	-17.33	+0.30	+0.72
	720	+12.80	+16.89	+18.54	-16.34	+3.62	-8.06	+10.61	-16.53	+0.06	+0.38
PatchTST	96	-4.13	+4.16	-12.82	-9.15	+2.16	-23.13	-16.94	-12.70	+0.89	-0.80
	192	-3.15	+0.22	-9.58	-9.52	+1.74	-20.75	-14.01	-10.55	+0.27	-0.46
	336	-0.51	+0.30	-7.94	-9.17	+1.52	-20.73	-13.21	-9.54	+0.17	-0.29
	720	-5.98	+0.45	-13.12	-7.27	+1.30	-16.27	-15.46	-9.18	+0.13	-0.15
iTransformer	96	-7.85	+0.19	-253.75	-10.94	+1.72	-27.63	-30.65	+0.97	+0.09	-0.02
	192	-6.69	-0.53	-237.80	-9.91	+1.44	-26.00	-24.01	+1.00	+0.10	-0.21
	336	-5.73	-0.14	-230.88	-11.02	+1.33	-26.71	-23.54	+0.62	+0.03	-0.44
	720	-7.30	+1.46	-213.41	-11.51	+1.19	-21.63	-24.91	+0.32	+0.09	O
TimeMixer	96	-4.32	+8.85	-1.81	O	+3.08	-15.37	-9.99	-41.87	+0.68	+1.50
	192	O	+5.03	+4.59	O	+2.90	-11.54	O	O	+0.41	+0.84
	336	O	+4.65	+3.46	O	+2.55	-10.25	O	O	+0.49	+1.08
	720	O	+1.76	+2.81	O	+2.37	-9.10	O	O	+0.41	O

E.2 LONG-LOOKBACK ROBUSTNESS AND RESOURCE FEASIBILITY

The controlled $L = 720$ track changes the input history while retaining the six datasets, four backbones, four prediction horizons, and seed 2021. It is a controlled extension when that lookback was not reported by a method source. Under the same hardware setup as the primary matrix, 934/1,056 jobs complete. Eight bare cells report CUDA OOM, making their 80 paired plugin cells unavailable; 33 additional plugin cells report direct CUDA OOM, and one KUP-BI cell fails for another execution reason. PIR completes 80/96 cells, with eight direct OOM and eight unavailable because the paired bare model OOMs. These outcomes remain explicit in Table 31 and are never encoded as negative accuracy gains. The OOMs are resource outcomes at the locked batch sizes on 24-GB RTX 3090 GPUs. Reducing training batch size changes optimization; reducing an online-update batch may change the update or label-release schedule. Either change defines a separate operating point and requires a separately recorded paired rerun. PIR's eight direct OOMs occur while constructing the device retrieval index, so reducing forecast batch size would not resolve that allocation.

Table 31: $L = 720$ **completion summary.** Counts are jobs in the six-dataset, four-backbone, four-horizon, seed-2021 track. Categories below the completed rows are mutually exclusive.

Outcome	Count
Planned jobs	1056
Completed jobs	934
bare / plugin	88 / 846
Bare OOM cells	8
Plugin cells unavailable: paired bare OOM	80
Direct plugin OOM	33
Other direct plugin failure	1
PIR: done / direct OOM / paired-bare OOM	80 / 8 / 8

Figures 7–11 show the paired gains for every completed $L = 720$ context. Gray cells distinguish a paired bare-model OOM (B), a plugin OOM (O), and another plugin failure (F); none is assigned an accuracy gain. Online-revision results retain their separate forecasting protocol.

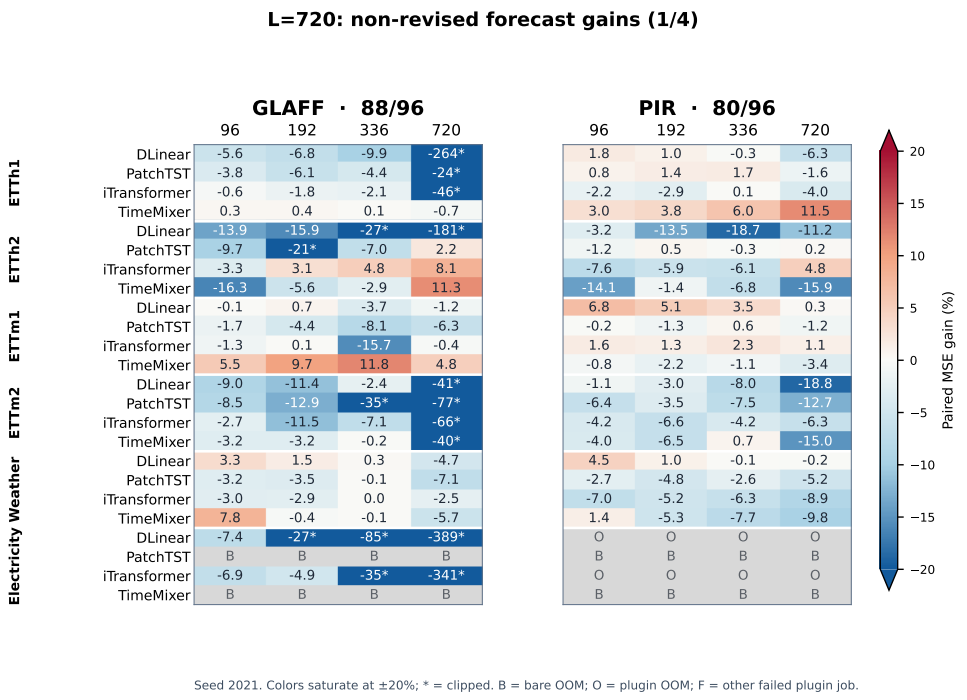

Figure 7: Controlled $L = 720$ paired MSE gains for GLAFF and PIR (seed 2021). GLAFF here uses release-CLI commit `4dedf10e00`, distinct from the $L = 96$ paper-snapshot matrix. Rows identify datasets and backbones; columns identify prediction horizons. Colors saturate at $\pm 20\%$, while printed values retain observed magnitudes. B/O/F mark distinct unavailable outcomes.

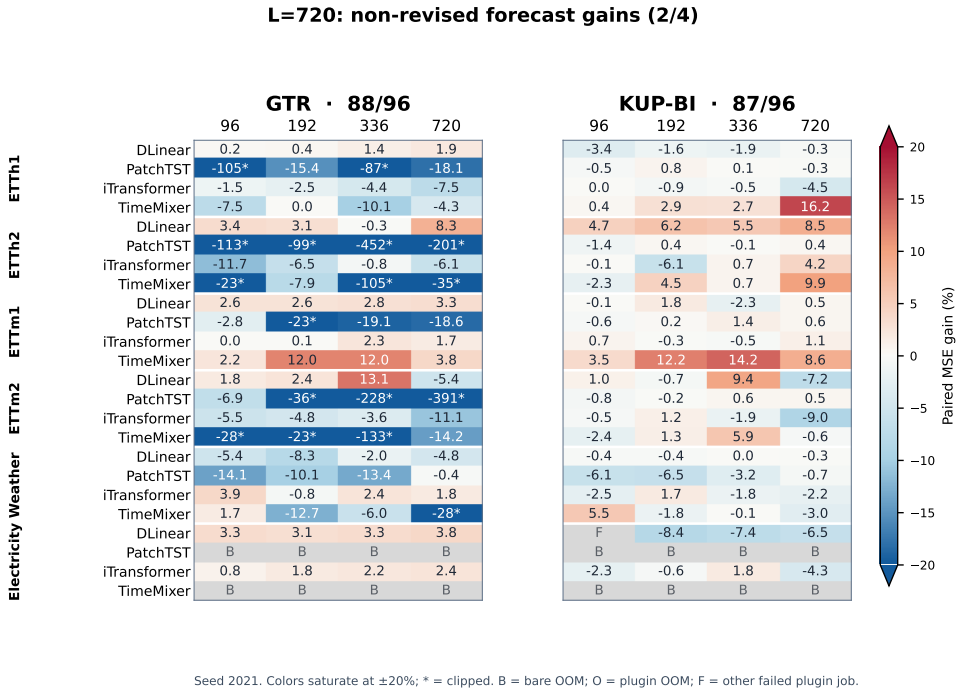

Figure 8: Controlled $L = 720$ paired MSE gains for GTR and KUP-BI (seed 2021), using the scale and status codes of Figure 7.

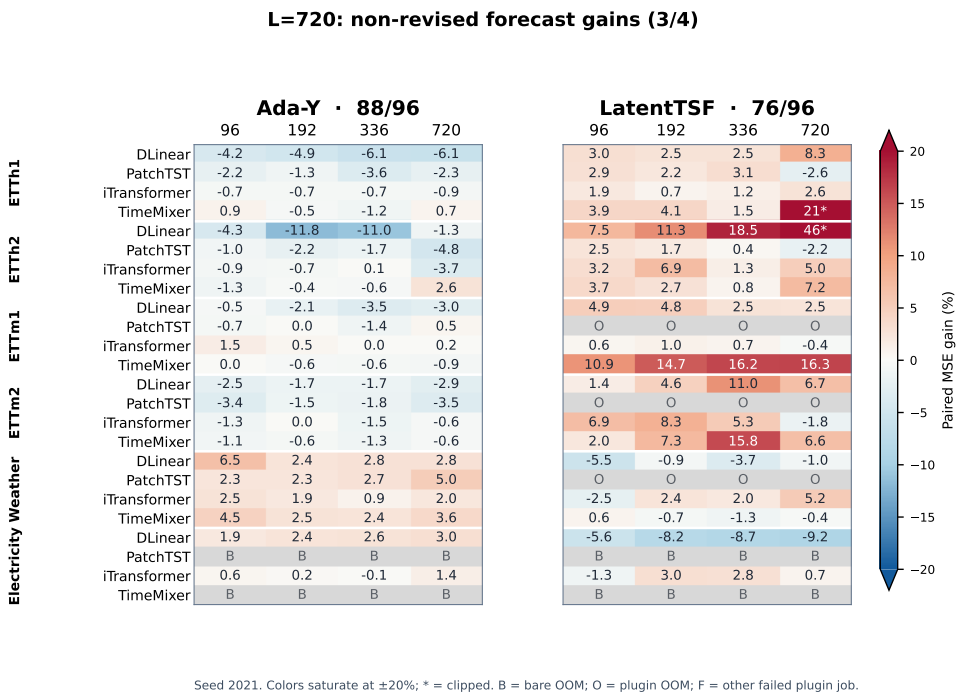

Figure 9: Controlled $L = 720$ paired MSE gains for Ada-Y and LatentTSF (seed 2021).

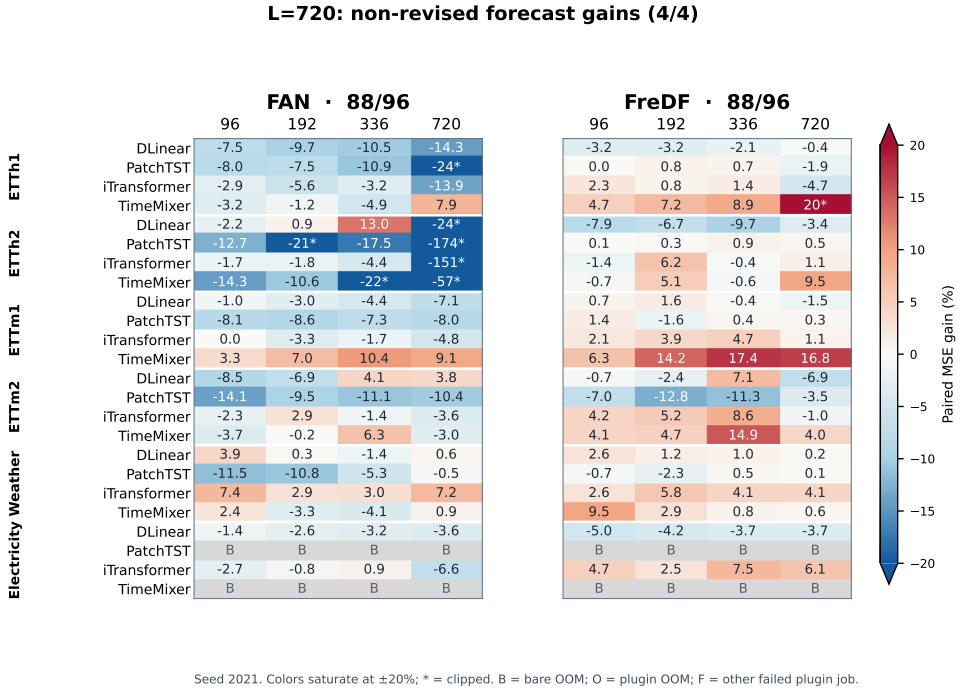

Figure 10: Controlled $L = 720$ paired MSE gains for FAN and FreDF (seed 2021).

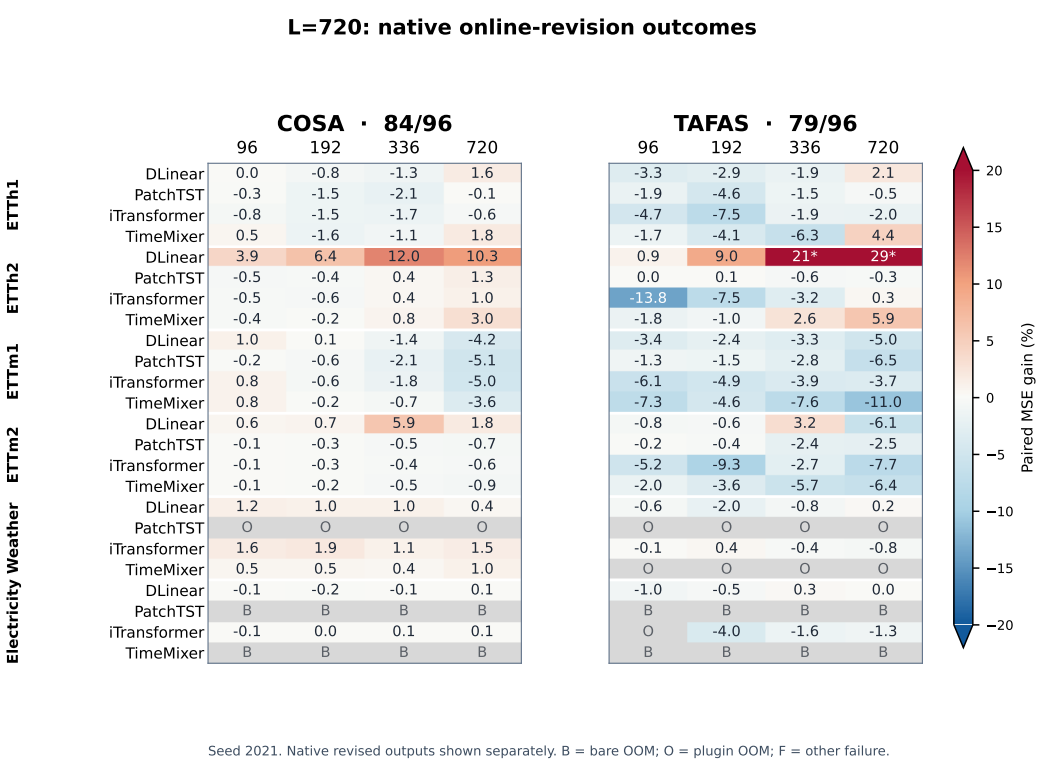

Figure 11: $L = 720$ paired MSE gains for COSA and TAFAS under their native online-revision protocols (seed 2021). These revised forecasts are displayed separately from the non-revised predictions in Figures 7–10.

Table 32: **Historical matched-lookback analysis (seed 2021).** Paired MSE gain is computed against the corresponding bare system. Each row contains only dataset–backbone–horizon cells completed at both lookbacks. GLAFF uses the release-CLI version at both lengths; its $L = 96$ values differ from the final three-seed paper-snapshot matrix. Native online-revision methods are separate. Sign agreement is descriptive.

Method	Contract	n	Median gain (%)		Positive (%)		Same sign (%)
			$L = 96$	$L = 720$	$L = 96$	$L = 720$	
GLAFF	Non-revised	88	-10.43	-3.72	9.1	20.5	70.5
PIR	Non-revised	80	+2.35	-2.18	78.8	32.5	51.2
GTR	Non-revised	88	-1.10	-3.96	45.5	39.8	67.0
KUP-BI	Non-revised	87	-0.28	-0.09	34.5	43.7	49.4
Ada-Y	Non-revised	88	+0.52	-0.64	58.0	38.6	71.6
LatentTSF	Non-revised	76	+1.86	+2.52	72.4	77.6	78.9
FAN	Non-revised	88	-1.62	-3.29	39.8	25.0	64.8
FreDF	Non-revised	88	-1.31	+0.77	30.7	63.6	55.7
COSA	Online revision	84	+0.18	-0.11	59.5	44.0	58.3
TAFAS	Online revision	79	-0.38	-1.90	38.0	19.0	58.2
All non-revised forecast	Emit once	683	-0.31	-0.63	45.2	42.2	63.5

The heatmaps show every $L = 720$ cell; the accompanying result ledger retains its exact gain and paired MSE. B denotes a paired bare-model CUDA OOM, O a plugin CUDA OOM, and F another plugin failure; none denotes zero gain. Table 31 gives the corresponding totals.

For a like-for-like historical input-length comparison, Table 32 pairs the archived seed-2021 $L = 96$ and $L = 720$ runs. GLAFF uses the release-CLI version in both columns, so this table does not

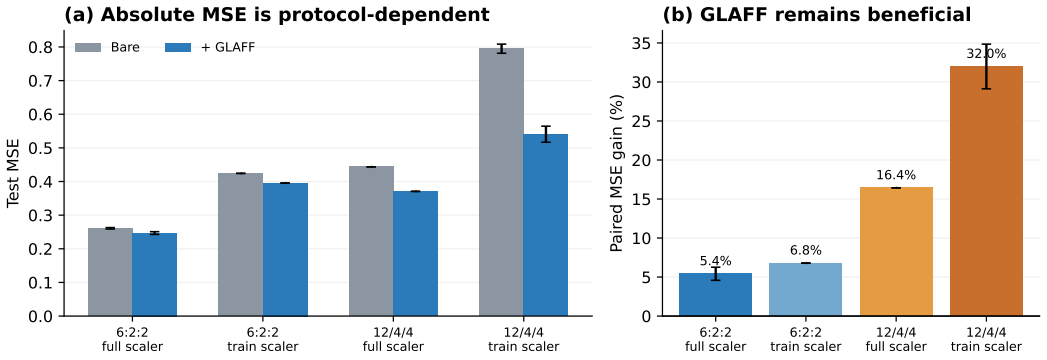

Figure 12: GLAFF protocol-sensitivity analysis on ETTh2/DLinear/$H = 720$. Error bars show sample standard deviations over three seeds for the released and controlled primary protocols; crossed split/scaler combinations are seed-42 diagnostics. Panel (a) shows that absolute normalized-space MSE changes with the evaluation protocol. Panel (b) reports only within-protocol paired gains.

compare its $L = 720$ run with the final paper-snapshot main result. Among 683 completed non-revised paired cells, 249 change gain sign; 29 of these have both gains within $\pm 1\%$. This describes sensitivity to the tested input lengths in that cohort. COSA and TAFAS retain online forecast revision and are shown separately. No $L = 720$ result is used as a clean-latency, checkpoint-replacement, or contract-preserving transformation measurement.

F GLAFF Protocol and Release-Version Bridge

The final $L = 96$ matrix uses GLAFF's paper-snapshot architecture and objective with the benchmark's train-only scaling and shared backbone recipes. This is distinct from the author's original data protocol, the later release-CLI defaults used in the separate $L = 720$ track, and the current HEAD. We compare the paper snapshot under the author's original and our controlled data protocols, then compare it with current HEAD under the same controlled protocol. Absolute normalized errors are not compared across different data protocols.

Table 33: Per-seed GLAFF protocol and version bridge on ETTh2/DLinear ($L = 96, H = 720$). The two crossed paper-snapshot split/scaler combinations are descriptive seed-42 diagnostics. Current HEAD is compared with the paper snapshot only under the identical controlled protocol; the releases differ in objective, architecture settings, and preprocessing.

Release	Split	Scaler fit	Seed	Bare	+GLAFF	Gain (%)
Paper snapshot	6:2:2	full series	42	0.2634	0.2514	4.56
			43	0.2594	0.2452	5.47
			44	0.2599	0.2437	6.23
	6:2:2	train only	42	0.4242	0.3954	6.79
	ETT 12/4/4	full series	42	0.4436	0.3707	16.43
	ETT 12/4/4	train only	42	0.7841	0.5532	29.46
			43	0.7905	0.5131	35.09
			44	0.8101	0.5556	31.42
Current HEAD	ETT 12/4/4	train only	42	0.7841	1.0269	−30.97
			43	0.7905	1.0235	−29.48
			44	0.8101	1.9605	−142.01

The released setting uses the generic 6:2:2 split. Its scaler is fitted to the complete 17,420-row series before splitting, and metrics are computed directly in that normalized space. Because validation and test observations contribute to these normalization statistics, this preprocessing uses future information under the benchmark's train-only scaling rule. We retain it only for the author-setting comparison and exclude it from the controlled matrix. The controlled setting uses the ETT-specific 12/4/4-month boundaries (8,640/2,880/2,880 rows), fits scaling statistics on the training rows only, and supplies the preceding L observations as history at validation and test boundaries. Consequently,

the two rows differ in training coverage, target interval, forecast timestamps, normalization statistics, and normalized channel weighting.

The two crossed seed-42 rows were preregistered to expose these coupled effects, not to select a preferred test result. They do not yield an additive causal decomposition: changing the scaler also changes the optimization objective and validation-selected checkpoint. The three-seed paper-snapshot comparison establishes only that this released path is beneficial in the ETTh2/DLinear/$H = 720$ context under both protocols. It does not establish protocol invariance over other datasets, horizons, or backbones. The released full-series scaler is retained for source fidelity but excluded from the train-only controlled benchmark.

The same-data-protocol bridge adds current HEAD seeds 42–44. With the same ETT split, train-only scaler, data file, L/H, and paired bare values, the paper snapshot obtains $+31.99\% \pm 2.86$ percentage points, whereas current HEAD obtains $-67.48\% \pm 64.54$ percentage points. Current HEAD simultaneously changes the mapper dimensions, dropout, forecast objective, and preprocessing path. The comparison therefore documents release identity and seed sensitivity; it neither attributes the sign change to one revision nor supports a method-family verdict. Table 33 reports every seed.

To determine which implementation choices must be preserved in the controlled integration, a separate seed-42 mechanism study modified the loss, fusion, and backbone fitting one at a time. Final-forecast-only training produced MSE 0.2725 against a paired bare MSE of 0.2634, whereas the native objective produced a positive paired gain. Fixed $50/50$ fusion retained only 0.65% gain, and freezing the paired backbone retained 6.49%. These controls support preserving the author loss and adaptive fusion; they do not prove that joint backbone updating is unimportant outside this cell.

G Deployment Size and Timing Requirements

Table 4 itemizes learned tensors and retrieval state rather than treating parameters as total storage. PIR state bytes sum normalized keys, memory moments, origins, and retained training values. For our uncached KUP-BI port, they sum host-resident keys, ratios, offsets, centered keys, and standard deviations; the fitting-only auxiliary cache is excluded because this port recomputes retrieval during test replay. The author runner instead precomputes and caches correlations for the test windows before forecasting, so its preprocessing, storage, and serving-time costs are not estimated by our KUP-BI row. Across the 96 PIR and 96 KUP-BI controlled cells, median device- and host-resident states are 90.43 and 701.24 MiB. Three fixed PIR cells spanning 20.5 MiB, 22.2 MiB, and 2.15 GiB of device state pass exact serialize/reload checks; their process peak RSS is 0.76, 0.77, and 5.54 GiB. These representative measurements are not extrapolated to every main-matrix cell.

Clean latency uses fresh processes, paired fixtures, matched batch size and precision, warm-up, repeated measurement, and explicit synchronization. All 768 forecast-once seed-2021 cells have same-source clean timing, including the recovered PIR and paper-snapshot GLAFF cells. Figure 3(b) uses the corresponding 96 contexts per method for gain, latency, and stored-state size. Ratios are formed within each cell before aggregation. Online revision streams require a different chronological workload and are not included in this forward/retrieval timing table. The per-cell timing and retrieval-state ledger accompanies the results.

The KUP-BI source precomputes candidate correlations for the full test set. To expose where its retrieval work occurs, Table 34 compares that cached path with our on-demand port on all four ETT datasets for DLinear and PatchTST at $L = 96, H = 96$, on the same host. Cache initialization includes parsing, scaling, training-library construction, test-window correlations, and disk writes; the disk column includes library and correlation files. The source then constructs auxiliaries from cached candidates, whereas our port builds a train-only index in 0.10–0.42 seconds and recomputes retrieval as each batch arrives. Across 32 prespecified, spread-out test windows per cell, auxiliary outputs differ by at most 1.7×10^{-6}. Each time is one full sweep, not a repeated clean-latency estimate. These auxiliary-stage measurements use different CPU/GPU execution paths and no backbone forward pass; they neither replace the full-system latency in Figure 3 nor estimate the author's published $L = 336$ setting. The source's offline cache also presumes the full test input series is available before the first forecast; this is an execution assumption, not evidence that a forecast uses its own future target.

Table 34: KUP-BI cached versus on-demand auxiliary construction on four ETT datasets ($L = 96, H = 96$). The same complete test-window sweep is timed in seconds; no backbone forward pass is included.

Dataset	Backbone	Windows	Cache init	Cached aux.	Stream aux.	Disk (MiB)
ETTh1	DLinear	2,785	2.92	0.87	3.66	88.0
ETTh1	PatchTST	2,785	2.99	0.89	2.06	88.0
ETTh2	DLinear	2,785	2.95	0.90	3.85	88.0
ETTh2	PatchTST	2,785	2.84	0.95	2.08	88.0
ETTm1	DLinear	11,425	44.14	4.15	183.57	360.7
ETTm1	PatchTST	11,425	44.39	4.12	18.82	360.7
ETTm2	DLinear	11,425	41.73	3.58	59.53	360.7
ETTm2	PatchTST	11,425	41.33	4.23	20.87	360.7

H INTERPRETATION LIMITS

All ten plugins have complete three-run coverage over the primary six-dataset $L = 96$ grid, but the $L = 720$ and exploratory Traffic tracks have one seed and include OOMs. Some pairings extend beyond author-reported settings. Checkpoint comparisons require a valid extraction and replacement interface, and the clean forward/retrieval timing does not cover online revision streams. Broader generalization needs more datasets, backbones, and input lengths.