TIME-SERIES INSTRUCTION FOLLOWING: AN ENDOGENOUS CONTROL LANGUAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series forecasters face changing phase, trend, and shared channel activity even when the prediction task stays fixed. We study whether this state can be expressed as instructions that remain useful when the predictor changes. Our *endogenous instruction language* separates message generation from forecasting response. A compiler maps the observed window and horizon to shared and optional channel-aligned numerical messages; architecture-specific readers learn to use them through joint forecast training. Across seven backbones, the value of instructions depends on the information accessible to the base prediction path. Capacity-matched comparisons establish the predictive value of multichannel content beyond local adaptation. Fixed-weight interventions show that trained predictors use the supplied content, and held-out probes recover phase, trend, and volatility from FITS messages. A frozen learned compiler generates messages for new windows while a new backbone and reader learn their forecasting response. Reusing a FITS compiler with DLinear reduces mean squared error by 12.66% relative to a frozen-random compiler on a new sensor dataset, with all five seeds improving. A horizon-aware DLinear reader also uses the ordered compiler message tokens generated by PatchTST, reducing MSE by 7.56% over a frozen-random compiler across five seeds. Native forecaster configurations also achieve lower mean squared error than official-code iTransformer reproductions across 12 datasets. These results identify learned state expression as a reusable part of forecasting knowledge.

1 INTRODUCTION

Instruction following lets large language models (LLMs) use task guidance alongside the content they process (Wei et al., 2022; Ouyang et al., 2022). Prompt and prefix tuning show that learned numerical vectors can also convey task-specific guidance to a frozen language model (Lester et al., 2021; Li & Liang, 2021). These approaches make guidance an explicit input to the model's response, suggesting a compiler–reader interface with a message-generating function and a separate reading operation. In forecasting, the task can stay fixed while the observed state changes, giving instructions a different role: expressing the conditions under which a prediction is made.

A multivariate forecaster observes a window $\mathbf{X} \in \mathbb{R}^{T \times C}$ of T time steps and C channels and predicts the next h steps. Phase within a recurring cycle, trend, variability, and shared channel activity provide context for this prediction. Forecasters already use statistics, periodic descriptors, global representations, and language prompts to expose such context (Zhou et al., 2025; Pei et al., 2025; Chi et al., 2024; Xue & Salim, 2023; Wang et al., 2025; Jin et al., 2024). Their predictive utility motivates a further question about the expression itself. Joint learning makes message content predictive with its reader; the next question is whether the learned state expression remains useful when the reader changes. *Can observed predictive state be written into messages whose generation remains fixed while different predictors learn to follow them?* That would make state expression reusable.

Multivariate forecasting exposes what such messages contribute. A channel-independent (CI) backbone predicts each channel from its own history (Nie et al., 2023; Zeng et al., 2022; Xu et al., 2024), so a shared message can expose observations outside its base prediction path. Direct mixers already learn channel interactions (Liu et al., 2024b; Wang et al., 2026), while restricted mixers use constrained operators such as Fourier filtering (Yi et al., 2024). The same guidance therefore reaches predictors with different access to shared state, and capacity-matched comparisons of local and multichannel guidance isolate the value of shared observations.

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

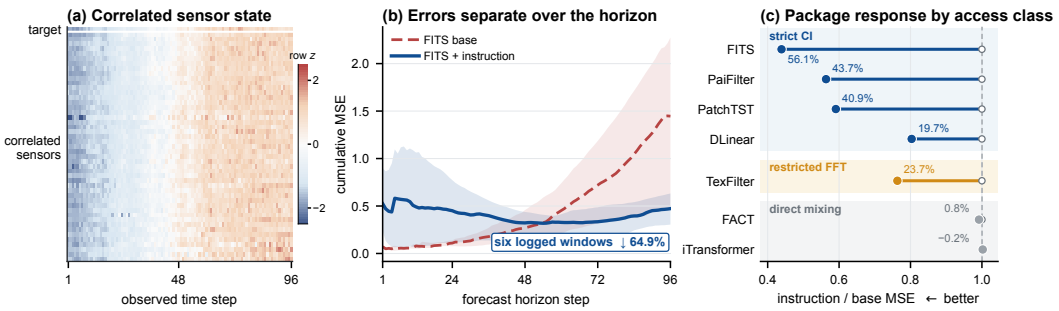


Figure 1: **From shared sensor structure to access-conditioned model response on PEMS04.** (a) An observed window for target sensor 306 and its 47 most correlated sensors, selected using training-split correlations; each row is window-standardized, and blue and red mark values below and above the row mean. (b) Median cumulative FITS MSE over the forecast horizon for six windows, with 25th–75th percentile bands. Over these six windows, the complete compiler–message–reader package reduces mean endpoint error by 64.9%. (c) Five-seed instructed/base MSE ratios for seven matched backbones, where values below one denote lower instructed error. This contrast motivates Sec. 4; Table 1 isolates the multichannel increment.

To make the state expression reusable, its generation must be separable from the operations that read it. We implement a compiler–reader interface: a *compiler* writes shared window-level and optional channel-aligned vectors from the observed window and horizon. An architecture-specific *reader*, implemented as an adapter (Houlsby et al., 2019), uses these messages through attention, hidden-feature modulation, or output correction, according to the backbone's operations. The compiler and reader form an *instruction package* around the original predictor's *base path* (Fig. 2). Joint forecast training lets message content and its use develop together. The learned compiler generates a message for each target window while a new backbone and reader learn how to use it for forecasting.

On PEMS04, Cross-channel packages reduce five-seed mean squared error (MSE) by 41.51% beyond Local packages at equal capacity on PatchTST, establishing the value of shared observations. Fixed-weight interventions then show that trained PatchTST and FITS predictors use the supplied content (Table 1). Held-out readouts recover phase, trend, and volatility from FITS messages (Fig. 3), linking the useful input to measurable state information in its learned expression. Package responses vary with information access across seven backbones. Native configurations lower MSE than official-code iTransformer reruns on all 12 datasets (Table 3).

The learned state expression remains useful after replacing the predictor. Frozen FITS and DLinear compilers support reuse in both directions, and the FITS compiler also serves PaiFilter (Table 2). Target backbones and readers are trained afresh, isolating the predictive contribution of the transferred compiler. With DLinear as the target, the FITS compiler reduces mean MSE by 11.32% on PEMS04 and 12.66% on PEMS08, while a horizon-aware residual reader reuses PatchTST's ordered compiler message tokens with a 7.56% reduction over a frozen-random compiler. This five-seed comparison demonstrates a state-preserving cross-reader transfer route: the target reader can use the frozen expression when it retains the message's temporal state. The complementary FITS-to-DLinear geometry analysis shows that source message directions remain useful across sensor sets within the residual-reader family (Fig. A.2).

2 Related Work

Instructions and learned conditioning. Instruction tuning learns responses to task descriptions (Wei et al., 2022; Ouyang et al., 2022), while prompt and prefix tuning encode task-specific guidance in continuous vectors (Lester et al., 2021; Li & Liang, 2021). Conditioning can act directly on computation: feature-wise linear modulation (FiLM) scales and shifts hidden features, and HyperNetworks generate parameters (Perez et al., 2017; Ha et al., 2017). What we determine here is whether a learned numerical message can be separated from its reader, so that a new predictor learns its response with the compiler frozen under a matched frozen-random control (Sec. 5.5).

Context for time-series prediction. Statistics and periodic descriptors provide numerical forecasting context (Zhou et al., 2025; Pei et al., 2025), and language offers another representation (Xue & Salim, 2023; Sun et al., 2024; Liu et al., 2024a; Wang et al., 2025). Language-based time-series models also use pretrained representations, semantic guidance, and covariate information (Zhou

et al., 2023; Mao et al., 2026; Ding et al., 2025; Cui et al., 2026; Jin et al., 2024; Gruver et al., 2023), with multimodal language modeling extending to classification (Cheng et al., 2025), and external modalities supplying observations absent from the series (Zhang et al., 2025). Foundation forecasters learn priors across series (Ansari et al., 2024; Das et al., 2024; Rasul et al., 2023; Woo et al., 2024), and IAMTime conditions time-series tasks on instruction-structured demonstrations (Saha & Shmakov, 2026). The window-state compiler transfers to a fresh backbone and reader while frozen, testing whether another operation can use its learned expression.

Cross-channel information paths. CI models share a forecasting function across channels without learned communication between them (Nie et al., 2023; Zeng et al., 2022; Xu et al., 2024; Han et al., 2023). Mixing models expose shared observations through clustering and timestamps (Chen et al., 2024; Wang et al., 2024a), multilayer perceptrons and attention (Chen et al., 2023; Zhang & Yan, 2023), variable tokens, or two-dimensional convolutions (Liu et al., 2024b; Wang et al., 2024c; 2026). The closest settings to ours separate less: InjectTST supplies global information to a CI Transformer through self-contextual attention (Chi et al., 2024), C-LoRA introduces channel-aware low-rank adaptation across backbone families (Nie et al., 2024), and language-prompt forecasters compile and consume guidance within one model (Wang et al., 2025; Pei et al., 2025); each keeps generation and consumption inside the same trained system. Soft prompts likewise transfer across tasks under a frozen language model (Vu et al., 2022), which motivates asking whether a learned time-series message survives reader replacement. Shared observations and channel-specific parameters thus provide distinct forms of adaptation, which capacity-matched input comparisons isolate.

3 METHODS

To reuse learned state expression, we separate the function that generates messages from the operations that read them. Given $\mathbf{X} \in \mathbb{R}^{T\times C}$ and horizon h, the predictor estimates $\mathbf{Y} \in \mathbb{R}^{h\times C}$. The compiler produces $\mathcal{I} = (G, \mathcal{V})$, where $G = [\mathbf{g}_1, \ldots, \mathbf{g}_K]$ contains K width-d vectors shared across the window and optional $\mathcal{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_C]$ contains one vector per channel. We call $\mathcal{I}$ the message, G the shared message, and $\mathcal{V}$ the channel-aligned message. Shared vectors expose aggregate state, whereas aligned vectors retain channel identity; the predictive content carried by this message is the learned state expression.

The paired study uses a common profile compiler with architecture-specific readers across seven backbones, while native Dynamic Prompt iTransformer (DPT) integrates both roles in a complete forecaster (Fig. 2); dynamic vectors depend on the window and static vectors are learned constants. Here a reader is the learned module that consumes a message, an adapter is its implementation around a backbone, and a reading operation is the functional form realized by that reader.

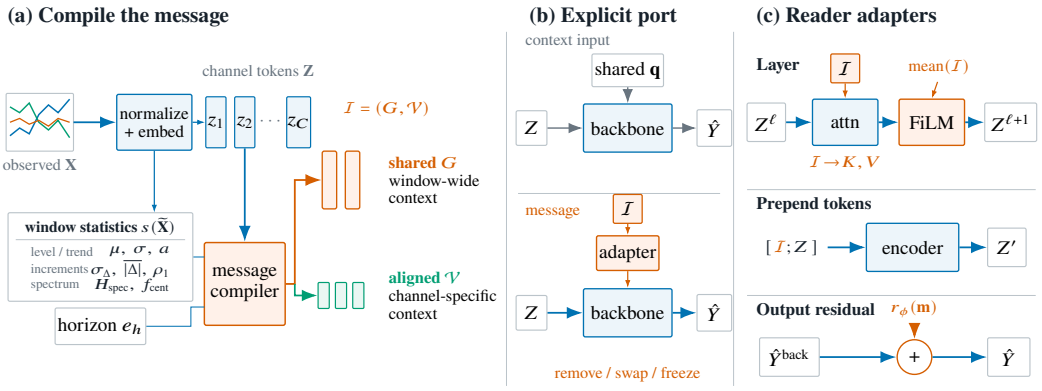


Figure 2: **One compiler–reader interface, architecture-specific readers.** (a) Native Dynamic Prompt iTransformer (DPT) maps the observed window and requested horizon to shared prompts G and optional channel-aligned prompts $\mathcal{V}$. (b) Shared context $\mathbf{q}$ enters through an explicit reader. (c) Architecture-specific readers consume the same message roles through attention keys/values with FiLM, prepended tokens, or additive output residuals.

3.1 COMPILER MESSAGE GENERATION

The paired compiler averages preprocessed channels into a length-T temporal profile $p(t)$, preserving temporal evolution independently of sensor count. A shared projection maps each of two equal segments to $\mathbb{R}^d$; layer normalization and training-time dropout yield two dynamic vectors in G. Channel averaging is permutation-invariant and retains aggregate temporal structure.

The horizon is fixed within each paired cell, so message variation comes from the observed profile. PatchTST and output-residual packages append two static vectors to G, giving $K = 4$. The paired iTransformer uses $K = 2$ and C aligned prompts in $\mathcal{V}$, combining each channel's token and statistics with the horizon. This preserves channel-specific descriptions alongside shared context (App. C.1).

3.2 ARCHITECTURE-SPECIFIC READERS

A common message format permits different reading operations. In DPT and the paired iTransformer package, G and $\mathcal{V}$ enter attention as keys and values, making their content available to channel queries. Channel queries read the message through attention keys and values. The same message also controls FiLM (Perez et al., 2017; Birnbaum et al., 2019). For $P = [G, \mathcal{V}]$, let $\bar{\mathbf{p}} = |P|^{-1} \sum_{\mathbf{p} \in P} \mathbf{p}$. A linear map converts this mean to $\gamma, \beta \in \mathbb{R}^d$, broadcast over the C channel tokens. These vectors scale and shift normalized post-attention states $\mathbf{u} \in \mathbb{R}^{C \times d}$:

$$\mathbf{u} \leftarrow \mathbf{u} \odot (1 + \tanh \gamma) + \beta. \tag{1}$$

PatchTST exposes shared context to each channel by prepending the ordered compiler message tokens G to its token sequence and discarding their outputs. DLinear, FITS, PaiFilter, TexFilter, and FACT read the message through an output residual: $\hat{\mathbf{Y}}^I = \hat{\mathbf{Y}}^{\text{back}} + \mathbf{1}_h r_\phi(\mathbf{m})^\top$. Here ϕ denotes reader parameters, $\mathbf{1}_h$ is an all-ones vector, and $\mathbf{m} = K^{-1} \sum_k \mathbf{g}_k$ is the pooled message. $r_\phi(\mathbf{m}) \in \mathbb{R}^C$ supplies one horizon-shared correction per channel. Each window-dependent correction adjusts forecast level across all steps; the backbone determines temporal shape. Here $\hat{\mathbf{Y}}^{\text{back}}$ is the instructed model's backbone output; the separately trained baseline is $\hat{\mathbf{Y}}^0$. For attention readers, the corresponding fixed-weight package-off output removes prompt inputs and bypasses FiLM (App. C.1).

The cross-reader transfer test retains the state expression when the target operation requires temporal structure. For the four-token PatchTST message $\mathbf{m}_b = [\mathbf{g}_{b1}, \ldots, \mathbf{g}_{b4}]$, a horizon-aware residual reader maps its ordered compiler message tokens to low-rank coefficients and emits

$$\Delta_{b,h,c} = \sum_{k=1}^{4} \sum_{r=1}^{16} u_{bkr} a_{cr} B_{kh}, \qquad \hat{\mathbf{Y}}^I_{b,h,c} = \hat{\mathbf{Y}}^{\text{back}}_{b,h,c} + \Delta_{b,h,c}, \tag{2}$$

where u_b is a learned projection of $\text{vec}(\mathbf{m}_b)$, a is a channel factor, and B contains four fixed-shape DCT-initialized horizon bases that remain trainable with the reader. This preserves token order and horizon-dependent state while keeping the target reader compact.

3.3 NATIVE FORECASTER

Native DPT makes shared and channel-aligned guidance available at every iTransformer layer. It embeds each normalized channel as $\mathbf{z}_c \in \mathbb{R}^d$, following the variable-token organization (Liu et al., 2024b), mixes these tokens with FlashAttention-2 (FA2) self-attention (Dao, 2024), and predicts $\hat{\mathbf{Y}}$ with a shared head. Backbone and compiler both read the normalized window $\widetilde{\mathbf{X}}$.

Native messages combine learned channel features, normalized-window statistics, and the requested horizon. For each channel, $\mathbf{s}_c(\widetilde{\mathbf{X}}) \in \mathbb{R}^8$ contains mean, standard deviation, least-squares trend slope, standard deviation of first differences, and mean absolute first difference. Lag-1 autocorrelation measures adjacent-step correlation; spectral entropy and spectral centroid summarize normalized power-spectrum entropy and power-weighted mean frequency. Channel averages $\mathbf{s} = C^{-1} \sum_c \mathbf{s}_c$ and $\bar{\mathbf{z}} = C^{-1} \sum_c \mathbf{z}_c$ describe the window collectively. A learned vector $\mathbf{e}_h$ identifies forecast length. A multilayer perceptron (MLP) with hidden width 256 compiles these inputs into shared prompts:

$$\mathbf{g}_k = f_{\text{comp},k}(\bar{\mathbf{z}}, \mathbf{s}, \mathbf{e}_h), \qquad \mathbf{g}_k \in \mathbb{R}^d, \quad k \in \{1, \ldots, K\}. \tag{3}$$

The aligned generator preserves the channel index alongside the shared message: $\mathbf{v}_c = \text{LN}(f_{\text{var}}([\mathbf{z}_c, \mathbf{s}_c, \mathbf{e}_h]))$, with layer normalization LN. PEMS08 uses the profile compiler for G while retaining this aligned generator and reader. Appendix D specifies the dataset-dependent native configurations.

3.4 TRAINING OBJECTIVE

The message has no prescribed linguistic labels. Forecast MSE learns its content and use by jointly training compiler, backbone, and reader. Optional fidelity supervision decodes a specified window descriptor; a determinant-based diversity loss encourages distinct shared DPT prompts:

$$\mathcal{L} = \mathcal{L}_{\text{forecast}} + \lambda_{\text{fid}}\mathcal{L}_{\text{fid}} + \lambda_{\text{div}}\mathcal{L}_{\text{div}}, \tag{4}$$

with zero weights disabling the corresponding terms. A future-window fidelity target is a training label; inputs remain the observed window and requested horizon. The fidelity decoder is unused at inference. Appendices B and D define the targets and losses. The matched-capacity and compiler-reuse packages learn message content directly from forecast MSE, so their transfer results measure forecasting state learned from the observed window and forecast labels.

3.5 COMPILER REUSE

Separating expression from response allows message generation to remain fixed while another predictor learns to read it. We copy a source profile compiler and its static vectors, freeze their parameters, and train a fresh target backbone and residual reader on target forecast labels. The fixed compiler produces new messages for observed target windows, and source preprocessing is retained (App. H.1). Because the shared message shape is independent of channel count, the target reader can map it to a different sensor set.

A frozen-random control matches compiler architecture, preprocessing, parameter counts, and target initialization; joint training from scratch provides the third condition (App. H.1).

The reader determines which parts of the state expression remain available: pooling the four tokens into one horizon-shared correction tests scalar reuse, whereas Eq. 2 tests reuse of the ordered temporal state. The targeted cross-operation comparison freezes learned or random PatchTST compilers and trains the DLinear reader from the same initialization over five paired seeds (App. H.1).

4 CROSS-CHANNEL STATE IN THE COMPILER–READER INTERFACE

A message provides another route from the window to the forecast, but its value depends on what the base path already observes and uses. The relevant comparison is therefore between information available to the base path and information added through the compiler. Cross-channel state makes this testable: the task stays fixed while the compiler can expose information beyond a channel's own history.

4.1 FROM OBSERVATION ACCESS TO A TESTABLE EXPECTATION

A CI backbone applies a shared prediction function independently to each channel. For channel c, the base and instructed paths are

$$\hat{\mathbf{y}}_c^0 = F_{\theta_0}(\mathbf{x}_c), \qquad \hat{\mathbf{y}}_c^I = F_{\theta_I,\phi}^I\big(\mathbf{x}_c; g_\psi(\mathbf{X}, h)\big), \tag{5}$$

where $\mathbf{x}_c$ is channel history. The backbone parameters θ_0 and θ_I are fitted separately; ϕ and ψ are reader and compiler parameters. At inference, the base path uses $\mathbf{x}_c$, whereas the instructed path additionally receives information from $\mathbf{x}_{c'}$ through g_ψ. A shared message can therefore help when it retains predictive state unavailable from local history. PatchTST, DLinear, FITS, and PaiFilter have this CI structure (Nie et al., 2023; Zeng et al., 2022; Xu et al., 2024; Yi et al., 2024).

The same message plays a different role in a mixing backbone. FACT and iTransformer already learn channel interactions through $H_\theta(\mathbf{X})$ (Wang et al., 2026; Liu et al., 2024b), while TexFilter connects channels through constrained channel-axis Fourier filtering (Yi et al., 2024). Here the message supplies another representation of observations already accessible to the base path. Access creates an opportunity; capacity-matched comparisons test the value of compiled state.

4.2 SEPARATING PACKAGE, INPUT-SCOPE, AND REUSE EFFECTS

Adding an instruction package changes both the computation and the information reaching it. For backbone b, dataset D, and a fixed recipe, let $M_{b,D}^0$ and $M_{b,D}^I$ denote five-seed mean MSE for the base and instructed packages. Their relative difference measures the total effect:

$$\Delta_I(b, D) = \frac{M_{b,D}^I - M_{b,D}^0}{M_{b,D}^0}, \tag{6}$$

where negative values favor the instructed predictor.
To identify the value of shared observations, let M^L and M^X denote errors of capacity-matched Local and Cross-channel packages. The input-scope gain $1 - M^X/M^L$ compares their information, whereas $1 - M^L/M^0$ measures local adaptation. The full-package gain $1 - M^X/M^0$ includes both changes. Reuse gain isolates the contribution of source learning to a fixed compiler. It compares frozen learned with frozen random compilers at the same target recipe, initialization, and trainable capacity. Five paired seeds quantify variability; Appendix C.2 formalizes the residual-headroom comparison used in the analysis.

5 Experiments: When and How Compiler Messages Help

We first measure the value of shared observations, then identify how a trained reader uses message content and finally test whether a frozen compiler transfers its state expression to new predictors and reading operations.

Data and evaluation. We use 12 datasets. Electricity Transformer Temperature (ETT) records transformer load and oil temperature (Zhou et al., 2021; Wu et al., 2021); ECL, Solar, and Traffic record client electricity demand, solar generation, and freeway occupancy (Lai et al., 2018). Weather contains meteorological measurements from TSLib (Wang et al., 2024b), and the Caltrans Performance Measurement System (PEMS) datasets PEMS03/04/07/08 record detector traffic flow (Song et al., 2020). Appendix A specifies signals and chronological splits. MSE and mean absolute error (MAE) average forecast errors; five seeds (2021–2025) yield mean$\pm$sample standard deviation (SD).

Paired protocol. The comparison unit is a fixed backbone recipe: separately trained base and package-enabled runs share the backbone architecture, split, optimizer, and training settings, with input and forecast lengths 96 and message width 128 (Table A.2). Local and Cross-channel packages have matched capacity (App. H). Native settings are dataset-specific (width 512 except ETT; App. D). PEMS horizons $\{12, 24, 48, 96\}$ (primary: 96) appear in App. E.1.

5.1 Package response across backbone access classes

Package response follows the access distinction on PEMS04: large gains occur in CI and restricted-mixing backbones (Table A.15). All 25 paired seed comparisons for the four CI models and TexFilter favor the package-enabled models. Mean MSE reductions span 19.7%–56.1% for CI and reach 23.7% for TexFilter (Table A.17). Direct mixers FACT and iTransformer have relative MSE changes of -0.8% and $+0.2\%$, with paired 90% bootstrap intervals $[-2.1, +0.3]\%$ and $[-0.5, +0.9]\%$ that include zero (App. E).
This contrast spans distinct reading operations. The CI group contains PaiFilter, FITS, PatchTST, and DLinear; each model trains the common profile-compiler design with its own backbone. PaiFilter and TexFilter are FilterNet's plain and contextual frequency filters (Yi et al., 2024). PatchTST reads prepended compiler message tokens, iTransformer reads keys/values with FiLM, and the remaining models read output residuals (App. C.1). TexFilter's restricted Fourier route is active: perturbing one channel changes other channels' outputs. The observed contrast is therefore consistent with the access expectation across both token and residual readers.

5.2 Multichannel input beyond local conditioning

At equal capacity, widening the compiler's input from one channel to the channel-averaged window moves PatchTST MSE from 0.44495 to 0.26024, a 41.51% improvement with 95% interval $[40.80, 42.22]\%$ and the same direction in all five seeds. Adding the local package reduces baseline MSE from 0.44674 to 0.44495 (Table 1(a)); shared observations supply the main increment.
C-LoRA RMLP (Nie et al., 2024) assigns channel-specific low-rank parameters while reading each channel independently; its local instruction package gives a 35.78% gain over the base. Multichannel input then gives a 0.99% local-to-multichannel gain with interval $[0.33, 1.57]\%$. Shared observations complement parameter personalization here, whereas they dominate the PatchTST contrast. FITS and DLinear gain a further 3.9% and 8.0% (App. H).
Package ablations distinguish window-dependent guidance from input-independent parameters. Static-only packages change PatchTST, DLinear, and FITS MSE by at most 1.1%, against full-package reductions of 40.9%, 19.7%, and 56.1%. On FITS, disabling fidelity changes MSE by -0.02%; retaining only the no-instruction adapter bias changes it by -0.09% (Table A.3). Dynamic conditioning drives the package response; input scope identifies the shared-observation effect (App. B).

Table 1: **Instruction content and forecasts on PEMS04.** Five-seed MSE, mean±sample SD. (a) Full-test, capacity-matched Local/Cross-channel models use the forecast-MSE package; PatchTST/RMLP use App. H. (b,c) Fixed-weight interventions, 134 windows. All/8/1 denote profile scope; Phase and Shift 24 use same-phase and preceding sampled donors, respectively. Panel (c): matched-swap ΔMSE.

(a) Capacity-matched trained input scope

Backbone	Base	Local	Cross-channel	Cross-channel vs. Local
FITS	1.1562±.0021	0.5279±.0041	0.5073±.0016	**−3.9%**
DLinear	0.4285±.0004	0.3740±.0010	0.3439±.0005	**−8.0%**
PatchTST	0.4467±.0040	0.4449±.0060	0.2602±.0014	**−41.5%**
C-LoRA RMLP	0.4133±.0046	0.2654±.0024	0.2628±.0027	**−1.0%**

(b) Frozen profile-context intervention

Backbone	Base	All	8 ch.	1 ch.	Phase	Shift 24
FITS	1.141±.002	**0.495±.003**	0.534±.004	0.843±.022	0.547±.003	0.831±.015
PatchTST	0.438±.006	**0.262±.003**	0.333±.010	1.070±.042	0.432±.005	0.916±.015

(c) Matched component substitution (ΔMSE)

Backbone	Level	Trend	Res. scale	Temp. shape
FITS	+2.25%	+1.17%	+0.86%	**+5.51%**
PatchTST	+0.00%	+0.41%	+1.31%	**+177.54%**

5.3 Context use by a fixed predictor

The trained input-scope comparison establishes the value of shared observations. Fixed-weight interventions next identify the content used by a trained predictor. In Table 1(b,c), compiler inputs change while backbone inputs, weights, static message vectors, and the 134 evaluated windows remain fixed. Eight channels retain 94.0% of the FITS gain and 59.7% of the PatchTST gain. One channel retains 46.2% for FITS, while PatchTST MSE becomes 1.070 against its 0.438 base. A donor window at the same daily phase gives MSE 0.547 for FITS and 0.432 for PatchTST. FITS thus tolerates phase-matched context, whereas PatchTST depends more on the current window.

Temporal shape is the most consequential tested profile component. Panel (c) writes $p(t) = \mu + a\tau(t) + \sigma r(t)$, separating mean level μ, trend slope a along centered time τ, residual root-mean-square scale σ, and zero-mean, detrended, unit-scale shape r. Replacing one component with a donor matched on the others gives the largest MSE increases for temporal shape: 5.51% for FITS and 177.54% for PatchTST. FITS also responds to level; PatchTST removes that component through per-channel normalization before pooling. For residual readers, the message adds a window-dependent, horizon-shared correction; App. B.2 reports its step-wise effect.

5.4 Window state readable from the message

Content interventions identify forecast dependence; readouts next identify the state represented in the message. For FITS (seed 2021), we analyze the pooled 128-dimensional mean of two dynamic and two static vectors. Principal component analysis (PCA) of 134 pooled messages finds 67.8%/25.3% of variation in its first two components; Fig. 3(a) colors their projection by daily-phase cosine.

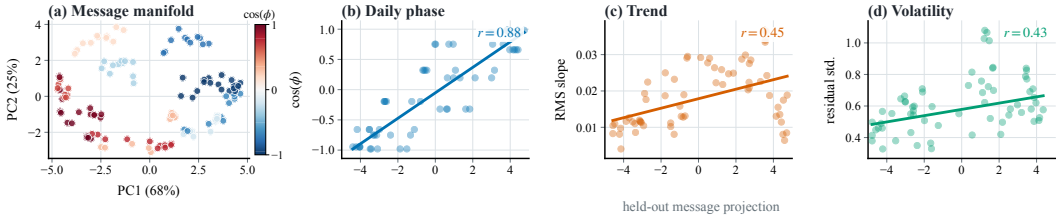


Figure 3: **Low-dimensional message geometry and held-out linear readouts on FITS/PEMS04.** (a) PCA of 134 messages, colored by daily-phase cosine. PC1/PC2 explain 67.8%/25.3% of variation. (b–d) Disjoint fit/evaluation histories span 67/64 windows after three boundary exclusions (App. B.2).

On 64 held-out windows, message–state readouts achieve $r = 0.88/0.45/0.43$ for phase, trend, and volatility; definitions are in App. B.2. Phase remains readable without time or phase tokens at the compiler input, and the output-correction norm correlates with per-window MSE reduction at $r = 0.41$.

5.5 COMPILER REUSE ACROSS PREDICTORS AND DATASETS

Window-state expression learned with one predictor can remain useful to another. Table 2 freezes PEMS04 source compilers and trains fresh target backbones and residual readers; five transfer settings compare frozen learned, frozen random, and jointly trained-from-scratch compilers over five paired seeds. Each frozen compiler has 6,913 parameters, including static vectors. The learned/random contrast holds preprocessing, target initialization, data order, and optimization fixed, so the measured difference is attributable to the transferred state expression.

Table 2: **Frozen-compiler reuse across predictors and datasets.** All source compilers are trained on PEMS04. (a) Target MSE is mean±sample SD over five paired seeds. Reuse gain compares learned and random compilers with both frozen; joint scratch trains all parameters. Targets are trained afresh under the matched recipes. † uses a token-preserving, horizon-aware residual reader. (b) Absolute MSE reduction with 95% paired-seed bootstrap intervals; positive values favor learned compilers; bold marks five-seed agreement.

(a) Frozen-compiler reuse and joint-training reference

Source → target	Data	Frozen random	Frozen learned	Gain ↑	Joint scratch
FITS → DLinear	PEMS04	0.3816 ± 0.0015	0.3384 ± 0.0010	$\mathbf{11.32}\%$	0.3433 ± 0.0019
DLinear → FITS	PEMS04	0.5697 ± 0.0029	0.5189 ± 0.0013	$\mathbf{8.92}\%$	0.5083 ± 0.0038
FITS → PaiFilter	PEMS04	0.3292 ± 0.0050	0.3182 ± 0.0035	$\mathbf{3.34}\%$	0.3145 ± 0.0015
PatchTST → DLinear†	PEMS04	0.2990 ± 0.0033	0.2764 ± 0.0038	$\mathbf{7.56}\%$	$--$
FITS → DLinear	PEMS08	0.5167 ± 0.0020	0.4513 ± 0.0009	$\mathbf{12.66}\%$	0.4625 ± 0.0018

(b) Paired reuse benefit and seed consistency

Source → target / data	Mean reduction	95% interval	Improving seeds
FITS → DLinear / PEMS04	0.04321	$[0.04141, 0.04471]$	5/5
DLinear → FITS / PEMS04	0.05083	$[0.04804, 0.05343]$	5/5
FITS → PaiFilter / PEMS04	0.01098	$[0.00921, 0.01251]$	5/5
PatchTST → DLinear† / PEMS04	0.02261	$[0.01856, 0.02602]$	5/5
FITS → DLinear / PEMS08	0.06541	$[0.06366, 0.06701]$	5/5

One learned compiler can support several target readers. On PEMS04, FITS-to-DLinear reuse reduces MSE by 11.32% relative to frozen random features, and the reverse direction gives 8.92%. The FITS compiler also improves PaiFilter by 3.34%, with all three contrasts improving in every seed. These direct residual-reader transfers establish predictive value across source and target backbones. The cross-operation comparison then tests whether the same state expression remains useful when the reader changes.

PatchTST compiles a window-normalized temporal profile into four ordered tokens. A DLinear reader that preserves this state type and emits a horizon-dependent residual lowers MSE from 0.29899 for the frozen-random compiler to 0.27637 for the frozen learned compiler, a 7.56% reduction with improvement in all five seeds and paired absolute interval $[0.01856, 0.02602]$ (Table 2). This result demonstrates a state-preserving cross-reader route: the target reader retains the temporal state carried by the message and exposes it through a horizon-dependent correction (App. H.1).

The FITS compiler also supplies useful guidance when both backbone and sensor set change. Trained on 307-channel PEMS04, it remains frozen as DLinear learns on 170-channel PEMS08 with target-split scaling and target forecast labels. Mean MSE falls from 0.51674 to 0.45133, a 12.66% reduction against a frozen-random compiler, with all five seeds improving. It is also 2.42% below joint training from scratch. All four beneficial reuse settings have positive paired-seed 95% intervals and lower mean MAE than frozen random compilers (Table 2(b)); App. H.1 reports seeds and costs.

Source-fitted geometry connects the transferred expression to the target reader's response: two PCA directions retain $91.0 \pm 0.3\%$ of PEMS08 test-message variance, and source-plane reconstructions preserve $R^2 = 0.847 \pm 0.008$ of the target-reader response (App. H.1).

5.6 INPUT VALUE ACROSS DATASETS

Input-scope gains depend on the dataset: Solar has higher mean absolute channel correlation than PEMS04, $\bar{r} = 0.914$ versus 0.770 (Table A.16), yet its package MSE changes range from -3.2% to $+3.4\%$, while ECL changes range from -4.9% to $+2.4\%$ (App. E). At equal capacity on ECL, multichannel input gains 0.45% (95% interval $[0.19, 0.81]\%$, all five seeds), versus 2.92% for local adaptation. The full-package gain tracks the PEMS04 base-error gap to a mixing reference ($\rho = 0.90$); ECL and Solar form a distinct regime ($\rho = 0.03$; App. C.2).

Headroom and input scope define separate regimes. Rank correlations use the paired cells in App. C.2; $\rho(M^0, \text{benefit})$ is the base-error reference.

Cell subset	n	$\rho(G_{\text{iT}}, \text{benefit})$	$\rho(M^0, \text{benefit})$
All datasets	11	0.67	0.72
PEMS04 only	5	0.90	0.90
ECL/Solar only	6	0.03	-0.26

Native Interface Evaluation

As the final system test, we ask whether the interface remains effective when compiler, backbone, and reader form one native forecaster. Table 3 compares DPT messages and matched static vectors with official-code iTransformer reproductions: MSE is lower on all 12 datasets, MAE on 11, and all 32 common PEMS horizon–metric cells favor DPT (App. E.1).

Table 3: **Native-system comparison** across 12 datasets at horizon 96 (five-seed mean±sample SD). Dynamic = DPT compilation; Static = learned controls with the same reader. System ΔMSE compares each native configuration with our official-code iTransformer reproduction. Apps. D and E give configurations; ΔMSE<0 favors the native system.

Dataset	Form	**Native system**		Official-code repro.		ΔMSE ↓
		MSE	MAE	MSE	MAE	
PEMS08	Dynamic	0.1986 ± 0.0104	0.2664 ± 0.0007	0.3157 ± 0.0135	0.3341 ± 0.0095	$\mathbf{-37.1}\%$
PEMS07	Dynamic	0.1223 ± 0.0027	0.2239 ± 0.0049	0.1482 ± 0.0044	0.2544 ± 0.0051	$\mathbf{-17.5}\%$
PEMS04	Dynamic	0.1486 ± 0.0018	0.2557 ± 0.0010	0.1681 ± 0.0024	0.2801 ± 0.0022	$\mathbf{-11.6}\%$
ECL	Dynamic	0.1387 ± 0.0006	0.2344 ± 0.0006	0.1483 ± 0.0005	0.2398 ± 0.0004	$\mathbf{-6.5}\%$
Solar	Dynamic	0.1938 ± 0.0072	0.2404 ± 0.0036	0.2046 ± 0.0011	0.2372 ± 0.0012	$\mathbf{-5.3}\%$
Weather	Dynamic	0.1665 ± 0.0015	0.2104 ± 0.0013	0.1746 ± 0.0013	0.2144 ± 0.0017	$\mathbf{-4.6}\%$
ETTm1	Static	0.3284 ± 0.0041	0.3645 ± 0.0025	0.3429 ± 0.0017	0.3770 ± 0.0009	$\mathbf{-4.2}\%$
PEMS03	Dynamic	0.1692 ± 0.0034	0.2729 ± 0.0018	0.1744 ± 0.0027	0.2840 ± 0.0026	$\mathbf{-3.0}\%$
ETTm2	Static	0.1799 ± 0.0011	0.2629 ± 0.0027	0.1847 ± 0.0010	0.2704 ± 0.0019	$\mathbf{-2.6}\%$
ETTh2	Static	0.2951 ± 0.0021	0.3461 ± 0.0020	0.3004 ± 0.0008	0.3502 ± 0.0008	$\mathbf{-1.8}\%$
Traffic	Dynamic	0.3903 ± 0.0022	0.2598 ± 0.0005	0.3934 ± 0.0010	0.2688 ± 0.0004	$\mathbf{-0.8}\%$
ETTh1	Static	0.3829 ± 0.0013	0.4038 ± 0.0006	0.3854 ± 0.0009	0.4040 ± 0.0007	$\mathbf{-0.6}\%$

Reductions range from 0.6% (ETTh1) to 37.1% (PEMS08) in MSE, establishing system-level value. The matched controls, fixed-weight interventions, frozen reuse, and cross-reader transfer then identify its source and portability.

The native configurations use the same interface with static ETT vectors or window-dependent messages elsewhere (App. D). On PEMS04, swapping all eight window statistics across frozen test windows changes MSE by -0.010%, while replacing any one group changes it by at most 0.060%; a held-out token readout recovers six such statistics with $R^2 = 0.878$–0.994 (App. B.1 and Table A.5). Together with the fixed-weight interventions, these results locate the forecast-relevant state in learned content rather than in explicit descriptors.

Residual readers consume horizon-shared corrections; horizon-aware readers preserve ordered temporal state. PatchTST→DLinear tests whether this expression survives the change (step-wise effects in App. B.2).

6 Conclusion

Learned window-state expression remains useful when the predictor changes: the compiler writes numerical messages, while each reader determines how their state alters forecasts. Native evaluation establishes end-to-end value; matched controls and interventions identify the value and content of shared observations; frozen reuse and PatchTST→DLinear transfer show that the expression can cross predictors and reading operations when the reader preserves its ordered temporal state.

This makes state type a first-class research variable: the compiler supplies a reusable window-state expression, while the reader determines how it is consumed. The freeze–random-control–fresh-reader protocol therefore gives the community a direct way to study state-preserving transformations and readers across new backbones, domains, and sensor shifts.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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

Sawyer Birnbaum, Volodymyr Kuleshov, Zayd Enam, Pang Wei Koh, and Stefano Ermon. Temporal FiLM: Capturing long-range sequence dependencies with feature-wise modulations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/2afc4dfb14e55c6face649a1d0c1025b-Abstract.html.

Jialin Chen, Jan Eric Lenssen, Aosong Feng, Weihua Hu, Matthias Fey, Leandros Tassiulas, Jure Leskovec, and Rex Ying. From similarity to superiority: Channel clustering for time series forecasting, 2024. URL https://arxiv.org/abs/2404.01340. NeurIPS 2024.

Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecasting, 2023. URL https://arxiv.org/abs/2303.06053. TMLR 2023.

Mingyue Cheng, Yiheng Chen, Qi Liu, Zhiding Liu, and Yucong Luo. InstructTime: Advancing time series classification with multimodal language modeling, 2025. URL https://arxiv.org/abs/2403.12371. WSDM 2025.

Ce Chi, Xing Wang, Kexin Yang, Zhiyan Song, Di Jin, Lin Zhu, Chao Deng, and Junlan Feng. InjectTST: A transformer method of injecting global information into independent channels for long time series forecasting, 2024. URL https://arxiv.org/abs/2403.02814. arXiv preprint 2024.

Zhiqing Cui, Binwu Wang, Qingxiang Liu, Yeqiang Wang, Zhengyang Zhou, Yuxuan Liang, and Yang Wang. Augur: Modeling covariate causal associations in time series via large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 764–787, 2026. doi: 10.18653/v1/2026.acl-long.32. URL https://aclanthology.org/2026.acl-long.32/.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning, 2024. URL https://arxiv.org/abs/2307.08691. ICLR 2024.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167, 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Kuiye Ding, Fanda Fan, Yao Wang, Ruijie Jian, Xiaorui Wang, Luqi Gong, Yishan Jiang, Chunjie Luo, and Jianfeng Zhan. DualSG: A dual-stream explicit semantic-guided multivariate time series forecasting framework, 2025. URL https://arxiv.org/abs/2507.21830. ACM MM 2025.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2310.07820.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *5th International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Lu Han, Han-Jia Ye, and De-Chuan Zhan. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting, 2023. URL https://arxiv.org/abs/2304.05206.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *PMLR*, 2019. URL https://arxiv.org/abs/1902.00751.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.01728.

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

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 95–104, 2018. doi: 10.1145/3209978.3210006. URL https://doi.org/10.1145/3209978.3210006.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021. URL https://arxiv.org/abs/2104.08691. EMNLP 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021. URL https://arxiv.org/abs/2101.00190. ACL 2021.

Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. UniTime: A language-empowered unified model for cross-domain time series forecasting, 2024a. URL https://arxiv.org/abs/2310.09751. WWW 2024.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting, 2024b. URL https://arxiv.org/abs/2310.06625. ICLR 2024 (Spotlight).

Dongdong Mao, Shijia Li, Gaowei Zhang, Wei Wang, and Yi Wang. tsLLM-SG: Time series forecasting by fine-tuning LLMs with dynamic soft prompts and gated frequency transformation adapters. In *Proceedings of DASFAA*, volume 16538 of *Lecture Notes in Computer Science*, pp. 522–538, 2026. doi: 10.1007/978-981-92-0372-7_32. URL https://link.springer.com/chapter/10.1007/978-981-92-0372-7_32.

Tong Nie, Yuewen Mei, Guoyang Qin, Jian Sun, and Wei Ma. Channel-aware low-rank adaptation in time series forecasting. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024. doi: 10.1145/3627673.3679884. URL https://arxiv.org/abs/2407.17246.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. ICLR, 2023. URL https://arxiv.org/abs/2211.14730.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155. NeurIPS 2022.

Jinglei Pei, Yang Zhang, Ting Liu, Jingbin Yang, Qinghua Wu, and Kang Qin. ADTime: Adaptive multivariate time series forecasting using LLMs. *Machine Learning and Knowledge Extraction*, 7(2):35, 2025. doi: 10.3390/make7020035.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871. AAAI 2018.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghoshal, Rikesh Cooper, Gautam Bhardwaj, Julian De Stefani, and Yuriy Nevmyvaka. Lag-Llama: Towards foundation models for probabilistic time series forecasting, 2023. URL https://arxiv.org/abs/2310.08278.

Anish Saha and Konstantin Shmakov. A foundation model for instruction-conditioned in-context time series tasks, 2026. URL https://arxiv.org/abs/2603.22586. arXiv preprint 2026.

Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 914–921, 2020. doi: 10.1609/aaai.v34i01.5438. URL https://ojs.aaai.org/index.php/AAAI/article/view/5438.

Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. TEST: Text prototype aligned embedding to activate LLM's ability for time series, 2024. URL https://arxiv.org/abs/2308.08241. ICLR 2024.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022. URL https://arxiv.org/abs/2110.07904.

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, 2024a. URL https://arxiv.org/abs/2409.18696. NeurIPS 2024.

Huiqiang Wang, Jieming Shi, and Qing Li. FACT: Fine-grained across-variable convolution for multivariate time series forecasting. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=j3gNYqrHtl. ICLR 2026 (Poster).

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

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. TimeXer: Empowering transformers for time series forecasting with exogenous variables. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024c. URL https://arxiv.org/abs/2402.19072.

Zesen Wang, Lijuan Lan, and Yonggang Li. Time-Prompt: Integrated heterogeneous prompts for unlocking LLMs in time series forecasting, 2025. URL https://arxiv.org/abs/2506.17631.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022. URL https://arxiv.org/abs/2109.01652. ICLR 2022 (Oral).

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers, 2024. URL https://arxiv.org/abs/2402.02592. ICML 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2106.13008.

Zhijian Xu, Ailing Zeng, and Qiang Xu. FITS: Modeling time series with $10k$ parameters, 2024. URL https://arxiv.org/abs/2307.03756. ICLR 2024 (Spotlight).

Hao Xue and Flora D. Salim. PromptCast: A new prompt-based learning paradigm for time series forecasting, 2023. URL https://arxiv.org/abs/2210.08964. TKDE 2024.

Kun Yi, Jingru Fei, Qi Zhang, Hui He, Shufeng Hao, Defu Lian, and Wei Fan. FilterNet: Harnessing frequency filters for time series forecasting. NeurIPS, 2024. URL https://arxiv.org/abs/2411.01623.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022. URL https://arxiv.org/abs/2205.13504. AAAI 2023.

Xiyuan Zhang, Boran Han, Haoyang Fang, Abdul Fatir Ansari, Shuai Zhang, Danielle C. Maddix, Cuixiong Hu, Andrew Gordon Wilson, Michael W. Mahoney, Hao Wang, Yan Liu, Huzefa Rangwala, George Karypis, and Bernie Wang. When does multimodality lead to better time series forecasting? *arXiv preprint arXiv:2506.21611*, 2025. URL https://arxiv.org/abs/2506.21611.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vSVLM2j9eie. Notable Top 5%.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021. doi: 10.1609/aaai.v35i12.17325. URL https://ojs.aaai.org/index.php/AAAI/article/view/17325.

Shiqiao Zhou, Holger Schöner, Huanbo Lyu, Edouard Fouché, and Shuo Wang. BALM-TSF: Balanced multimodal alignment for LLM-based time series forecasting. *arXiv preprint arXiv:2509.00622*, 2025. URL https://arxiv.org/abs/2509.00622.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. doi: 10.52202/075280-1877. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/86c17de05579cde52025f9984e6e2ebb-Abstract-Conference.html.

A DATASETS

The datasets expose the forecasting interface to different forms of temporal state, from transformer loads to traffic flow. We forecast all channels on 12 datasets spanning six signal families. ETTh1/ETTh2 and ETTm1/ETTm2 record six load measurements and oil temperature from two electricity transformers, sampled hourly and every 15 minutes, respectively (Zhou et al., 2021; Wu et al., 2021). ECL contains hourly demand from 321 electricity clients, Solar contains 10-minute generation from 137 plants, and Traffic contains hourly freeway occupancy from 862 sensors (Lai et al., 2018). Weather provides 21 meteorological variables at 10-minute resolution in the TSLib benchmark (Wang et al., 2024b). PEMS03/04/07/08 contain 5-minute traffic flow from Caltrans Performance Measurement System detectors (Song et al., 2020).

Every forecast uses 96 observed steps. The primary horizon is 96; PEMS also uses horizons 12, 24, and 48. Chronological splits keep later observations out of training, and each scaler is fitted on the training segment and then held fixed. Table A.1 specifies the processed signals, sampling intervals, file lengths, channel counts, and target splits.

Table A.1: Dataset characteristics. T_{file} is the number of timestamps in the processed file, and C is the number of jointly forecast channels. ETT uses fixed 12/4/4-month train/validation/test ranges. The remaining ratios are chronological splits. PEMS04 and PEMS08 contain multiple raw features, and the loader selects traffic flow so that one channel corresponds to one detector.

Dataset	Forecast variables	Interval	T_{file}	C	Target split
ETTh1/ETTh2	Six transformer-load measurements and oil temperature	1 h	17,420	7	12/4/4 months
ETTm1/ETTm2	Six transformer-load measurements and oil temperature	15 min	69,680	7	12/4/4 months
ECL	Client-level electricity load	1 h	26,304	321	70/10/20%
Solar	Plant-level solar generation	10 min	52,560	137	70/10/20%
Weather	Meteorological measurements	10 min	52,696	21	70/10/20%
Traffic	Freeway occupancy	1 h	17,544	862	70/10/20%
PEMS03	Detector traffic flow	5 min	26,208	358	60/20/20%
PEMS04	Detector traffic flow	5 min	16,992	307	60/20/20%
PEMS07	Detector traffic flow	5 min	28,224	883	60/20/20%
PEMS08	Detector traffic flow	5 min	17,856	170	60/20/20%

A.1 WITHIN-BACKBONE PAIRED PROTOCOL

Paired comparisons measure what instructions add to a fixed predictor. Each pair uses the same architecture, split, optimizer, and training recipe, with separately trained parameters. Recipes appear in Table A.2; native-system settings are in Appendix D.

Table A.2: Paired protocol and backbone-specific recipes (Sec. 5). Heads/layers and feed-forward width (FFN) specify attention backbones; pat. is early-stopping patience and lr is learning rate. The type1 schedule halves lr each epoch. Counts are dynamic profile vectors plus channel-aligned vectors for iTransformer (iT), and profile plus static message vectors otherwise. The width-256 MLP and horizon embedding belong to native DPT and the aligned generator, not to the shared profile compiler.

Hyperparameter	Value	Hyperparameter	Value
Message width d	128	Dropout	0.1
Input/label/output	96 / 48 / 96	Seeds	2021–2025
Profile segment length	48	Profile projection	$48 \to 128$ + LayerNorm
PatchTST patch/stride	16 / 8	Message vectors	$2 + C$ (iT); $2 + 2$ (others)

Backbone	Dataset	Heads/layers	FFN	Batch	Epochs/pat.	lr	Schedule
FITS, DLinear, FACT	PEMS04	–	–	32	50/5	3×10^{-4}	cosine
iTransformer	All three	4/2	512	32	50/5	3×10^{-4}	cosine
PatchTST	PEMS04	16/4	512	32	80/15	10^{-3}	cosine
PatchTST	ECL, Solar	16/4	512	32	80/15	3×10^{-4}	cosine
PaiFilter, TexFilter	PEMS04	–	–	16	20/6	5×10^{-3}	type1
PaiFilter	ECL	–	–	16	20/6	3×10^{-4}	type1
PaiFilter	Solar	–	–	16	20/6	3×10^{-4}	cosine
DLinear	ECL, Solar	–	–	32	50/5	3×10^{-4}	cosine

B MECHANISM CONTROLS

A useful message must affect prediction through its content. The controls below distinguish that response from static parameters and auxiliary supervision, then ask what information the trained reader uses. Local and multichannel compilers separate adaptation from access to shared observations; frozen context substitutions identify the temporal information on which a forecast depends. The readouts then identify state information accessible in the message.

Auxiliary supervision constrains what a message represents during training. For a message summary $\bar{\mathbf{p}}$ and row-specific target q, fidelity is $\mathcal{L}_{\text{fid}} = \text{mean}[(\text{dec}(\bar{\mathbf{p}}) - q(\mathbf{X}, \mathbf{Y}))^2]$, averaged over examples and target coordinates. Profile-plus-static wrappers pool all prompts into a scalar linear decoder and have no diversity term. Native DPT normally pools G for an MLP decoder; segmented targets instead use one output per shared prompt. Its future-target loss divides coordinate-wise MSE by a running target variance clamped below at 10^{-4}. To encourage distinct shared prompts, define their cosine-similarity matrix $\mathbf{C}_{ij} = \langle \mathbf{g}_i, \mathbf{g}_j \rangle / (\|\mathbf{g}_i\| \|\mathbf{g}_j\|)$ and penalize $\mathcal{L}_{\text{div}} = -\log\det(\mathbf{C} + 10^{-3}\mathbf{I})$. These optional terms in Eq. 4 use the native targets in Table A.12. Table A.3 tests their removal. Both weights are zero in the trained input-scope controls.

Table A.3: Controls for the source of the instruction-package effect (Sec. 5.2). Seed counts and comparison cells are stated per row. The fidelity-off result uses two matched FITS seeds for the PEMS04 profile cell. ECL and Solar use a different input-profile target.

Control	Result (PEMS04 unless noted)	Interpretation
Fidelity off (λ_{fid}=0, 2 matched seeds)	0.5069 vs. instructed 0.5070 (5-seed instructed: 0.5074)	FITS gain persists without fidelity
Bias only, no instructions (5 seeds)	1.1551 vs. base 1.1562 (FITS)	content dependence exceeds the unconditional shift
Static control only	PatchTST 0.4409→0.4456 (5), DLinear 0.4285→0.4279 (5), FITS 1.1562→1.1543 (2)	the dynamic profile supplies the response beyond static control
Diversity off (λ_{div}=0, 5 seeds)	0.1890 vs. profile 0.1895 (iTransformer cell)	-0.29% paired-mean change
Learning-rate control (FITS, Weather)	lr 10^{-4}/5×10^{-4}: base 0.1965/0.1953, instructed 0.1940/0.1933	improvement persists at both rates

B.1 NATIVE STATISTIC-GROUP INTERVENTION

The native compiler receives learned tokens alongside eight explicit statistics. Replacing the statistics while keeping the tokens intact tests whether these descriptors affect its forecast. On PEMS04, a deterministic permutation assigns every test window a different donor. The backbone input, horizon, weights, channel identities, and non-target statistics stay fixed. We swap level/scale (mean and standard deviation), change (slope, first-difference standard deviation, and mean absolute first difference), dependence (lag-1 autocorrelation), or spectrum (spectral entropy and centroid).

Across five seeds, the largest absolute mean MSE change is 0.060%; replacing all eight statistics changes MSE by -0.010% (Table A.4). The native PEMS04 forecast is therefore largely insensitive to the explicit statistics under this intervention. The compiler still receives unchanged-window tokens; the next readout tests whether they retain the same state information.

Statistical information retained in the tokens. Using the same five frozen checkpoints, we fit readouts from the channel tokens before compilation to the eight statistics that the compiler receives. Both are computed from the RevIN-normalized window (Kim et al., 2022); no explicit statistic enters the readout input. A shared channel readout maps $\mathbf{z}_c$ to $\mathbf{s}_c$, while a separate pooled readout maps $\bar{\mathbf{z}}$ to $\mathbf{s}$. Linear ridge regression is compared with a joint eight-output MLP with a 64-unit GELU hidden layer.

Table A.4: Frozen statistic-group shuffle for native DPT on PEMS04 (seeds 2021–2025). Negative ΔMSE denotes lower error after shuffling. No group increases MSE under this intervention.

Shuffled statistics	MSE mean±SD	ΔMSE vs. intact
None (intact)	0.148623 ± 0.001844	0.000%
Level/scale	0.148623 ± 0.001843	-0.0001%
Change	0.148534 ± 0.001906	-0.060%
Dependence	0.148608 ± 0.001851	-0.010%
Spectrum	0.148595 ± 0.001854	-0.019%
All eight	0.148608 ± 0.001936	-0.010%

The chronological 60/20/20 split supplies 106/35/35 disjoint length-96 histories at stride 96. Channel readouts have 32,542/10,745/10,745 window–channel rows; all channels of a window remain in the same split. Data scaling and readout input/target standardization fit on training data. Validation selects the ridge penalty per target from nine values between 10^{-6} and 10^{2}. The MLP uses AdamW with learning rate and weight decay both 10^{-3}, batch size 1,024, and at most 500 epochs with patience 40. Its checkpoint minimizes validation MSE averaged over all eight standardized targets. Test $R^2 = 1 - \mathrm{MSE}/\operatorname{Var}(s)$ is reported in Table A.5. The shuffle uses 3,208 overlapping forecast windows from the same test partition.

Table A.5: Statistics readable from native DPT tokens on PEMS04 (five-checkpoint mean±sample SD of held-out R^2). Each MLP jointly predicts eight statistics. Channel and pooled readouts correspond to the aligned and shared compiler inputs, respectively.

	Channel token		Pooled token	
Statistic	Ridge	MLP	Ridge	MLP
Trend slope	0.926 ± 0.008	0.994 ± 0.001	0.987 ± 0.005	0.879 ± 0.058
First-difference SD	0.462 ± 0.009	0.924 ± 0.004	0.731 ± 0.055	0.112 ± 0.431
Mean absolute first difference	0.450 ± 0.008	0.902 ± 0.004	0.727 ± 0.077	0.185 ± 0.133
Lag-1 autocorrelation	0.266 ± 0.006	0.882 ± 0.009	0.662 ± 0.110	-0.031 ± 0.249
Spectral entropy	0.146 ± 0.005	0.878 ± 0.010	-0.160 ± 0.463	0.752 ± 0.039
Spectral centroid	0.254 ± 0.005	0.903 ± 0.006	0.729 ± 0.090	0.150 ± 0.533
Mean	0.909 ± 0.002	0.992 ± 0.003	0.030 ± 0.003	-0.672 ± 0.528
Standard deviation	0.408 ± 0.034	0.969 ± 0.007	-0.063 ± 0.004	-0.043 ± 0.171

The channel MLP recovers the six change, dependence, and spectrum statistics with mean R^2 between 0.878 and 0.994. Spectral entropy, for example, rises from 0.146 with ridge to 0.878 with the MLP. Pooled spectral entropy also becomes readable ($R^2 = 0.752$). Mean and standard deviation describe the RevIN-normalized window and primarily reflect learned channel-wise affine parameters and near-constant effects; the other six targets describe temporal variation.

Together with Table A.4, these results locate overlapping statistical information in the unchanged token path. Replacing the explicit descriptors disrupts their correspondence to the current window without removing the corresponding information from the tokens. The readouts measure information available in the token path; the shuffle measures forecast dependence on the explicit descriptors. Appendix B.2 measures dependence on the profile compiler's observed temporal input.

B.2 Cross-channel context intervention

For the profile compiler, the first question is whether shared observations add value beyond conditioning on the target channel. The local control compiles a dynamic profile from $\mathbf{x}_c$ and applies row c of the same $d \to C$ projection used by the multichannel adapter. The multichannel condition instead compiles the channel-averaged window. Both add 46,516 parameters, contain two profile and two static vectors, and disable future auxiliary supervision. The split, optimizer, stopping rule, and seeds are identical, so the comparison changes information access at fixed capacity.

Local conditioning accounts for most of the output-residual gain in these two backbones. Shared observations then lower MSE by a further 3.9% for FITS and 8.0% for DLinear, with improvement in all ten paired seeds.

Table A.6: Capacity-matched trained context control on PEMS04 (MSE, mean±SD over seeds 2021–2025). Base, Local, and Cross-channel conditions have identical trainable parameter counts. Only the dynamic profile's input scope changes. Cross-channel MSE is lower in all ten paired seeds.

Backbone	Added params	Base	Local	Cross-channel	Cross-channel vs. Local
FITS	46,516	1.1562±.0021	0.5279±.0041	0.5073±.0016	−3.9%
DLinear	46,516	0.4285±.0004	0.3740±.0010	0.3439±.0005	−8.0%

Training lets each reader adapt to its input scope. To identify the context used by an already trained reader, we instead change the profile input at frozen checkpoints. The FITS and PatchTST profile-plus-static models each combine two static control vectors with two instance-specific profile vectors. We evaluate 134 evenly spaced PEMS04 test windows for seeds 2021–2025, holding the CI backbone, adapter capacity, weights, and static vectors fixed. The encoder receives all channels from the correct window, eight evenly spaced channels, the first channel, a different window at the same daily phase, or the preceding sampled window (a cyclic 24-timestamp shift except at the boundary).

Table A.7: Frozen-checkpoint profile-context intervention on PEMS04 (MSE, mean±SD over five seeds). Base has no instruction package. All, 8 channels, and 1 channel specify the observed channels supplied to the profile compiler. Static message vectors, adapter capacity, weights, backbone inputs, and evaluated windows are fixed. Only the profile compiler's context changes. Eight channels retain 94.0%/59.7% of the FITS/PatchTST gain, while one channel retains 46.2% for FITS and gives PatchTST MSE 1.070 versus base 0.438.

Backbone	Base	All	8 channels	1 channel	Same phase	Shift 24
FITS	1.141 ± .002	0.495 ± .003	0.534 ± .004	0.843 ± .022	0.547 ± .003	0.831 ± .015
PatchTST	0.438 ± .006	0.262 ± .003	0.333 ± .010	1.070 ± .042	0.432 ± .005	0.916 ± .015

The trained readers use both the breadth and the timing of their context (Table A.7). Restricting the profile to one channel or substituting a wrong window raises MSE with the same parameters and weights. A phase-matched donor separates the readers further: FITS retains more of its benefit from another window at the same daily phase, whereas PatchTST relies more on the correct instance. The shared temporal profile carries this multichannel aggregate context. The data file is `figures/data/cross_channel_context.json`, with analysis in `figures/analyze_cross_channel_context.py`.

To identify which temporal information drives this response, we decompose the actual encoder input as $p(t) = \mu + a\tau(t) + \sigma r(t)$. Here τ is centered linear time, and r has zero mean, zero linear trend, and unit root-mean-square magnitude. For each component, the Hungarian algorithm assigns one-to-one donors that minimize distance over the other three components. Donor and target lookbacks cannot overlap. Eight principal components represent r during matching and retain 97.5% of its variation. We swap only the selected component, keeping the rest of the profile intact. Table A.8 gives the full MSE values behind Table 1(c).

Table A.8: Frozen matched profile-component intervention on PEMS04 (MSE, mean±SD over seeds 2021–2025 and 134 windows). Δ rows give change from the intact instructed checkpoint. FITS pools dataset-standardized channels. PatchTST pools after channel normalization.

Backbone	Metric	Intact	Level	Trend	Res. scale	Temp. shape
FITS	MSE	.4953±.0028	.5064±.0026	.5011±.0025	.4995±.0025	.5225±.0023
	Δ (%)	0.00	+2.25	+1.17	+0.86	+5.51
PatchTST	MSE	.2615±.0027	.2615±.0027	.2626±.0023	.2649±.0022	.7258±.0206
	Δ (%)	0.00	+0.00	+0.41	+1.31	+177.54

Temporal shape produces the largest response for both readers. FITS also uses level, trend, and residual scale from its dataset-standardized profile, without per-window normalization. PatchTST normalizes channels before pooling, making level constant and concentrating its response on shape. Preprocessing determines the available state. Individual substitutions measure reliance on each component at fixed weights.

All substitutions retain the backbone input, weights, static message vectors, and non-target components. One-to-one matching uses only non-target components and non-overlapping look-backs, with reconstruction error below 2.4×10^{-7}. The script and five-seed records are in `experiments/profile_component_shuffle_20260827/`. The forecast-step analysis relates the residual reading operation to where prediction improves; message readouts then identify the encoded state.

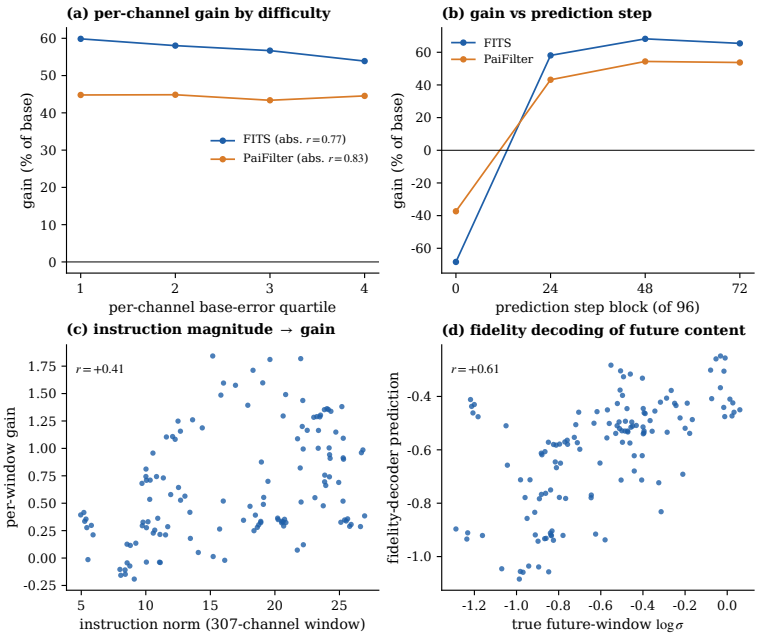


Figure A.1: Modulation surface of the instructed PEMS04 cell (134 test windows, seed 2021, both CI backbones). (a) Relative per-channel gain varies little across quartiles of the per-channel base error (FITS 59.9%→53.9%, PaiFilter 44.8%→44.5%), while absolute MSE reduction correlates with base error at $r = +0.77/+0.83$. (b) A horizon-shared correction produces $+43 \ldots 68\%$ reductions on steps 24–96; the first 24 steps show -68% (FITS) and -37% (PaiFilter) changes. The net error reduction concentrates later, where the base model has larger errors. (c) Output-correction norm $\|\mathbf{b}\|_2$ vs. per-window MSE reduction, $r = +0.41$; message magnitude here denotes the decoded correction. (d) Fidelity-decoder prediction vs. true future-window $\log \sigma$, $r = +0.61$.

A residual reader supplies one window-dependent, horizon-shared correction per channel across the forecast horizon. Its effect at each step depends on the backbone error at that step. For the instructed model's internal backbone error $e_{t,c} = \hat{Y}^{\text{back}}_{t,c} - Y_{t,c}$ and offset $b_c = r_\phi(\mathbf{m})_c$, the squared-error reduction is $e^2_{t,c} - (e_{t,c} + b_c)^2 = -2b_c e_{t,c} - b_c^2$. The linear term measures alignment between the correction and backbone error, while the quadratic term penalizes correction magnitude. For the base-path error $e^0_{t,c}$, the full-package effect decomposes into backbone and reader contributions:

$$(e^0_{t,c})^2 - (e_{t,c} + b_c)^2 = \left[(e^0_{t,c})^2 - e^2_{t,c}\right] - 2b_c e_{t,c} - b_c^2.$$

Figure A.1(b) shows the resulting horizon profile: the first 24 steps change by -68% (FITS) and -37% (PaiFilter), while steps 24–96 improve by $+43 \ldots 68\%$. The shared offset therefore acts against a time-varying error profile; joint training also changes the backbone.

The representation analysis in Figure 3 asks what state is linearly accessible in those messages. The pooled 128-dimensional prompts have two principal components explaining 93.1% of variation, with windows arranged by daily phase. A direction fitted on 67 windows correlates with phase cosine at $r = +0.88$ on 64 later windows. The compiler's phase association therefore comes from the observed profile. Separately fitted directions decode trend and volatility at $r = +0.45/+0.43$. The message thus contains a low-dimensional representation in which several forecasting states are accessible by linear projection.

The probe uses seed 2021, window length 96, and stride 24. After the first 67 fitting windows, three windows are omitted so that none of the 64 evaluation histories overlaps a fitting history; windows within each split remain overlapping. For centered fit messages Q and state labels s, the direction is $w = Q^\top s / \|Q^\top s\|_2$, a covariance-direction projection rather than a 128-parameter least-squares fit. Phase is the cosine of the start index modulo 288 daily samples. Trend is the root-mean-square of channel-wise, per-step least-squares slopes. Volatility is the standard deviation over all channel–time entries after subtracting each channel's fitted slope times the time index. Labels use only observed-window values or the start index. Fitting windows determine directions and calibration lines; held-out windows measure their correlation with state.

C Adapter mapping and headroom sensitivity

Adapter insertion points specify how each backbone reads the message. Residual headroom then relates the package response to the error left by that predictor.

C.1 Cross-backbone adapter mapping

All paired-study models compress the complete window into two shared, instance-specific profile vectors. The profile-plus-static package (run label *SP*) appends two learned static vectors to G. The paired iTransformer instead adds C channel-aligned vectors, combining each channel's token and statistics with the horizon. Table A.9 links these packages to their insertion points; Table A.15 reports the resulting forecast response. K/V means attention keys/values; FFT means fast Fourier transform.

Table A.9: How each backbone reads the instruction package. The base arm removes the complete listed package and adapter while retaining every backbone block.

Backbone(s)	Access	Package	Adapter
Paired iTransformer	mixing	2 shared profile + C aligned	per-layer K/V prompts plus package-mean FiLM
PatchTST	CI	2 profile + 2 static	prepend tokens and discard their output states
DLinear, FITS, PaiFilter	CI	2 profile + 2 static	horizon-shared, channel-specific output residual
TexFilter	mixing (FFT)	2 profile + 2 static	horizon-shared, channel-specific output residual
FACT	mixing	2 profile + 2 static	horizon-shared, channel-specific output residual

CI and mixing refer to information access in the retained backbone, before the reader is added. TexFilter, for example, mixes channels through its FFT path: perturbing one input channel changes other output channels in the functional test recorded in `figures/data/lineage_implementation_audit.json`. This classification fixes the predictor's existing access before evaluating the benefit of the added message. The baseline and instructed models train separately. In a fixed attention model, removing prompt tokens and bypassing FiLM yields an instruction-off computation with the same backbone weights, including the loss of prompt attention.

C.2 Residual-headroom sensitivity

Residual headroom compares a backbone's base error with a mixing reference on the same dataset. For backbone b, dataset D, and reference r, define

$$G_r(b, D) = \frac{M^0_{b,D}}{M^0_{r,D}} - 1. \tag{7}$$

A value of zero matches the reference MSE. The complementary data-level statistic is $\bar{r} = \mathrm{mean}_{i<j} |\rho_{ij}|$, the mean absolute training-set Pearson correlation between channels.

The package-response analysis uses iTransformer as the mixing reference; PEMS04 also provides FACT for all five non-reference backbones. Changing the reference rescales headroom but preserves

its order, because every base MSE is divided by the same positive number (Table A.10). The rank correlation with package benefit is consequently $\rho = 0.90$ under either reference, as implied by the common-denominator normalization. The matched scope table reports the corresponding input-scope gain.

Table A.10: Residual headroom for the five analyzed non-reference backbones against both available five-seed mixing references on PEMS04. TexFilter is the FFT mixer, and the other four are CI. Reference MSE is 0.1891 for iTransformer and 0.1182 for FACT.

PEMS04 backbone	Base MSE	$100G_{\mathrm{iT}}$	$100G_{\mathrm{FACT}}$	MSE reduction
TexFilter (FFT mix)	0.2317	22.5	96.1	23.7%
DLinear	0.4285	126.6	262.6	19.7%
PatchTST	0.4409	133.2	273.1	40.9%
PaiFilter	0.5460	188.7	362.0	43.7%
FITS	1.1562	511.5	878.5	56.1%

Across datasets, the five PEMS04 cells give $\rho = 0.90$, while the six lower-headroom ECL/Solar cells give $\rho = 0.03$; pooling all eleven gives $\rho = 0.67$ (Table A.11). Headroom therefore describes the PEMS04 package ordering here; ECL and Solar form a separate regime.

Table A.11: Headroom–benefit rank correlations across dataset subsets. The eleven paired cells are analyzed with `figures/analyze_gap_sensitivity.py`.

Cell subset	n	$\rho(G_{\mathrm{iT}}, \text{benefit})$	$\rho(M^0, \text{benefit})$
All three datasets	11	0.67	0.72
PEMS04 only	5	0.90	0.90
Leave PEMS04 out	6	0.03	−0.26
Leave ECL out	8	0.83	0.83
Leave Solar out	8	0.74	0.90

D PER-ROW INSTRUCTION VARIANTS

The complete-system configurations instantiate two distinct control forms. ETT uses input-independent learned vectors, so its future-log-volatility loss regularizes a dataset-level shared control state. Weather generates messages dynamically with the same target. Traffic predicts one volatility estimate per forecast segment, while ECL and Solar reconstruct an input-window profile. PEMS uses dataset-specific configurations. Table A.12 lists forms and settings. Settings use d/FFN for message/feed-forward widths and lr/bs/ep/pat for learning rate/batch size/epochs/patience.

Table A.12: Control forms and configurations for Table 3.

Dataset	Form	Config
ETTh1/2, ETTm1/2	static	tuned: d=128, 8 heads, FFN 2048, per-dataset lr (ETTh1 d=64, 3 layers, ETTh2 4 layers, ETTm1/2 2 layers)
Weather	dynamic	tuned: d=512, 8 heads, 3 layers, lr 5×10^{-4}
Traffic	dynamic	tuned: d=512, 8 heads, 4 layers, FFN 2048, bs 16, 50 ep, lr 10^{-4}
PEMS03/04/07/08	dynamic	d=512, 8 heads, 4 layers, bs 16, 50 ep, pat 5. FFN/lr: 1024/10^{-4} (PEMS03), 1024/3$\times10^{-4}$ (PEMS04), 2048/3$\times10^{-4}$ (PEMS07), 2048/10^{-4} (PEMS08)
ECL, Solar	dynamic	d=512 recipe

PEMS08 uses the two-segment profile compiler for G; other dynamic native rows use the token/statistic MLP in Eq. 3. All retain the channel-aligned generator and attention/FiLM reader.

Prompt counts follow the same row-specific configuration: ETT, Traffic, and PEMS use K=2 shared prompts; Weather, ECL, and Solar use K=4. Every row adds C channel-indexed prompts, implemented as learned constants for static ETT and as shared-MLP outputs for dynamic rows. The counts and base arguments are in `benchmark/logs/flags.csv`.

E REPRODUCTION DETAILS

System comparisons use the native configurations in Table A.12 and official-code iTransformer configurations specified below. Within-backbone comparisons use the paired protocol in Appendix A.1. Both use the processed signals and chronological partitions in Table A.1, with training-fitted scalers.

The baselines in Tables 3 and A.14 are local reruns of the iTransformer release (Liu et al., 2024b), using its data pipeline and the configurations of the reported runs.

Table A.13 specifies the Traffic and PEMS baselines. Traffic uses eight heads, four layers, and batch 16; PEMS07/08 use batch 32. All PEMS baselines use `--use_norm 0`. Each model is trained with seeds 2021–2025. Full commands are in `experiments/official_multiseed/PROTOCOL_CERTIFICATE.md`.

Table A.13: Official-code baseline settings and PEMS evaluation windows. Native-system settings are given in Table A.12. Epochs/pat. denotes the training limit and early-stopping patience. Window counts are at horizon 96.

Dataset	Width	FFN	Learning rate	Epochs/pat.
Traffic	512	512	10^{-3}	10/3
PEMS03/07/08	512	512	10^{-3}	10/3
PEMS04	1024	1024	5×10^{-4}	10/3

Dataset	Official-code test windows	Native test windows
PEMS03	5,051	5,052
PEMS04	3,208	3,208
PEMS07	5,454	5,455
PEMS08	3,381	3,381

E.1 FULL-HORIZON PEMS RESULTS

Table A.14 evaluates four PEMS forecast horizons with the same pipeline and five-seed reproduction. DPT lowers both mean MSE and mean MAE in all 16 dataset–horizon comparisons.

Table A.14: PEMS results across four forecast horizons. DPT and the official-code reproduction both report five-seed mean±sample SD (2021–2025), and DPT has lower MSE and MAE in all 16 cells. Deltas use displayed means, and negative values favor DPT. For PEMS03 at $h = 12$, baseline seed-level MSE is 0.4452 in one run and 0.0718–0.0731 in the other four; all five enter the summary.

Dataset	h	DPT MSE	DPT MAE	Code-repro MSE	Code-repro MAE	ΔMSE	ΔMAE
PEMS03	12	0.0651 ± 0.0008	0.1685 ± 0.0018	0.1470 ± 0.1667	0.2405 ± 0.1418	$\mathbf{-55.7}\%$	$\mathbf{-29.9}\%$
PEMS03	24	0.0847 ± 0.0010	0.1922 ± 0.0021	0.0923 ± 0.0007	0.2006 ± 0.0014	$\mathbf{-8.2}\%$	$\mathbf{-4.2}\%$
PEMS03	48	0.1267 ± 0.0024	0.2336 ± 0.0034	0.1364 ± 0.0033	0.2464 ± 0.0030	$\mathbf{-7.1}\%$	$\mathbf{-5.2}\%$
PEMS03	96	0.1692 ± 0.0034	0.2729 ± 0.0018	0.1744 ± 0.0027	0.2840 ± 0.0026	$\mathbf{-3.0}\%$	$\mathbf{-3.9}\%$
PEMS04	12	0.0766 ± 0.0009	0.1797 ± 0.0023	0.0809 ± 0.0003	0.1881 ± 0.0005	$\mathbf{-5.3}\%$	$\mathbf{-4.5}\%$
PEMS04	24	0.0895 ± 0.0006	0.1947 ± 0.0016	0.1002 ± 0.0008	0.2123 ± 0.0011	$\mathbf{-10.7}\%$	$\mathbf{-8.3}\%$
PEMS04	48	0.1166 ± 0.0037	0.2252 ± 0.0067	0.1337 ± 0.0020	0.2469 ± 0.0015	$\mathbf{-12.8}\%$	$\mathbf{-8.8}\%$
PEMS04	96	0.1486 ± 0.0018	0.2557 ± 0.0010	0.1681 ± 0.0024	0.2801 ± 0.0022	$\mathbf{-11.6}\%$	$\mathbf{-8.7}\%$
PEMS07	12	0.0600 ± 0.0004	0.1556 ± 0.0014	0.0662 ± 0.0004	0.1638 ± 0.0003	$\mathbf{-9.4}\%$	$\mathbf{-5.0}\%$
PEMS07	24	0.0778 ± 0.0017	0.1742 ± 0.0037	0.0846 ± 0.0007	0.1860 ± 0.0012	$\mathbf{-8.0}\%$	$\mathbf{-6.3}\%$
PEMS07	48	0.0961 ± 0.0017	0.1954 ± 0.0031	0.1146 ± 0.0007	0.2198 ± 0.0010	$\mathbf{-16.1}\%$	$\mathbf{-11.1}\%$
PEMS07	96	0.1223 ± 0.0027	0.2239 ± 0.0049	0.1482 ± 0.0044	0.2544 ± 0.0051	$\mathbf{-17.5}\%$	$\mathbf{-12.0}\%$
PEMS08	12	0.0758 ± 0.0020	0.1748 ± 0.0052	0.1482 ± 0.0038	0.2009 ± 0.0049	$\mathbf{-48.9}\%$	$\mathbf{-13.0}\%$
PEMS08	24	0.0994 ± 0.0008	0.1948 ± 0.0034	0.1807 ± 0.0059	0.2316 ± 0.0064	$\mathbf{-45.0}\%$	$\mathbf{-15.9}\%$
PEMS08	48	0.1437 ± 0.0027	0.2307 ± 0.0053	0.2512 ± 0.0098	0.2864 ± 0.0070	$\mathbf{-42.8}\%$	$\mathbf{-19.4}\%$
PEMS08	96	0.1986 ± 0.0104	0.2664 ± 0.0007	0.3157 ± 0.0135	0.3341 ± 0.0095	$\mathbf{-37.1}\%$	$\mathbf{-20.3}\%$

E.2 CROSS-BACKBONE PAIRED RESULTS AND SUPPORTING STATISTICS

Within-backbone comparisons measure the instruction-package effect. Each cell in Table A.15 trains a reference *base* and an instructed counterpart with only the package added, using seeds 2021–2025. Package gains depend on both architecture and dataset, as the cell-level intervals below show.

Paired bootstrap 90% intervals quantify the smaller FACT/iTransformer responses. Using 10^5 seed-paired resamples with `figures/compute_bootstrap_cis.py`, the intervals are $[-2.1, +0.3]\%$ for FACT and $[-0.5, +0.9]\%$ for iTransformer on PEMS04, $[+1.8, +3.1]\%$ for iTransformer on ECL, and $[-2.5, +4.1]\%$ on Solar. The wider Solar interval accompanies base variation of $\pm 5.4\%$. These intervals describe package effects within each architecture–dataset cell.

Table A.15: Paired instruction-package MSE changes under backbone-specific recipes (d=128, App. A.1). All cells report five-seed mean±sample SD (2021–2025), rounded to four decimals so each change can be recomputed from the displayed pair. "(prof)" denotes a profile shared compiler with channel-aligned messages (App. C.1). $\bar{r}$ is the mean absolute Pearson correlation over channel pairs. ΔMSE= $(\text{instructed} - \text{base})/\text{base}$, with negative values denoting improvement. Bold marks the largest reduction per dataset group.

Backbone	Base MSE	With instructions	ΔMSE	Seeds
PaiFilter	0.5460 ± 0.0062	0.3076 ± 0.0014	-43.7%	5
PatchTST	0.4409 ± 0.0042	0.2606 ± 0.0024	-40.9%	5
TexFilter (FFT mix)	0.2317 ± 0.0019	0.1768 ± 0.0015	-23.7%	5
DLinear	0.4285 ± 0.0004	0.3442 ± 0.0008	-19.7%	5
FITS	1.1562 ± 0.0021	0.5074 ± 0.0015	-56.1%	5
FACT	0.1182 ± 0.0020	0.1172 ± 0.0015	-0.8%	5
iTransformer (prof)	0.1891 ± 0.0020	0.1895 ± 0.0012	$+0.2\%$	5
PatchTST (ECL)	0.1632 ± 0.0008	0.1552 ± 0.0010	-4.9%	5
DLinear (ECL)	0.1949 ± 0.0003	0.1887 ± 0.0007	-3.2%	5
PaiFilter (ECL)	0.1900 ± 0.0009	0.1882 ± 0.0010	-0.9%	5
iTransformer (ECL)	0.1402 ± 0.0005	0.1436 ± 0.0011	$+2.4\%$	5
PatchTST (Solar)	0.2037 ± 0.0047	0.2106 ± 0.0076	$+3.4\%$	5
DLinear (Solar)	0.2867 ± 0.0025	0.2794 ± 0.0025	-2.6%	5
PaiFilter (Solar)	0.2500 ± 0.0012	0.2419 ± 0.0020	-3.2%	5
iTransformer (Solar)	0.1990 ± 0.0108	0.2008 ± 0.0048	$+0.9\%$	5

Channel association provides a data-level contrast to backbone-specific headroom. We compute $\bar{r}$, the mean absolute Pearson correlation over channel pairs, from each training segment in Table A.1: 8,640 hourly or 34,560 quarter-hourly ETT rows, 60% of PEMS, and 70% of the other datasets. Every forecast channel is included, including all 137 columns in the headerless Solar file. The values in Table A.16 support the comparison in Section 4.

Table A.16: Training-set mean absolute channel correlation $\bar{r}$ on all 12 datasets, using every forecast channel. This describes the data, whereas residual headroom also depends on the backbone.

Dataset	$\bar{r}$	Dataset	$\bar{r}$
ETTh1	0.311	ECL	0.497
ETTh2	0.349	Solar	0.914
ETTm1	0.314	Weather	0.294
ETTm2	0.349	Traffic	0.579
PEMS03	0.841	PEMS08	0.792
PEMS04	0.770	PEMS07	0.806

F INFERENCE COST

The native interface shares compiler parameters across channels, but its readers still incur channel-dependent computation. For the PEMS04 seed-2021 checkpoint (d=512, four layers, FFN width 1024), the shared-prompt MLP, aligned-prompt MLP, horizon embedding, and prompt normalizations total 975,360 parameters. Four FiLM maps add 2,101,248, yielding 3,076,608 instruction parameters. The retained backbone has 8,512,710 parameters; the forecast-producing path totals 11,589,318. Thus instructions account for 26.5% of forecast-path parameters, excluding the 264,969 parameters in two auxiliary decoding heads. Counts vary with the dataset configuration.

Dynamic rows apply one shared aligned-prompt MLP to every channel instead of learning a $C \times d$ prompt table. Computing statistics and generating aligned tokens nevertheless scales linearly with C. At each layer, C channel queries read $K + C$ additional prompt keys and values, adding attention-score work of order $C(K + C)d$ plus key/value projections. Parameter sharing therefore controls compiler size while the reader cost continues to depend on channel count.

G PER-SEED RESULTS AND GAIN PROFILES

Seed-level MSE changes show whether a paired mean reflects a consistent response. All 50 runs in the ten improving non-reference cells reduce MSE (Table A.17). PEMS04 FACT/iTransformer changes straddle zero, while ECL iTransformer increases MSE in all five seeds ($+1.5\%$ to $+3.9\%$, mean $+2.4\%$). Solar iTransformer spans -6.9% to $+6.1\%$ (mean $+1.1\%$). These differences preserve the architecture- and dataset-specific interpretation of package value.

Table A.17: Per-seed relative MSE changes (%) for every paired cell (seeds 2021–2025). All 50 runs of the ten improving non-reference cells are negative. The PEMS04 FACT/iTransformer cells straddle zero, the ECL iTransformer cell is positive on all five seeds, and the Solar iTransformer cell spans both signs. Means are the arithmetic mean of the listed seeds, differing from Table A.15's change-of-rounded-means by ≤ 0.2 point (largest: Solar mixing, $+1.1\%$ vs. $+0.9\%$).

Backbone	Dataset	2021	2022	2023	2024	2025	Mean±SD
PaiFilter	PEMS04	−43.9	−44.7	−43.9	−42.5	−43.3	-43.7 ± 0.8
PatchTST	PEMS04	−41.0	−40.5	−41.7	−40.9	−40.5	-40.9 ± 0.5
TexFilter (FFT mix)	PEMS04	−24.3	−22.7	−24.1	−24.4	−22.8	-23.7 ± 0.8
DLinear	PEMS04	−19.7	−19.8	−19.5	−19.8	−19.5	-19.7 ± 0.2
FITS	PEMS04	−56.2	−56.1	−56.2	−56.1	−55.9	-56.1 ± 0.1
FACT	PEMS04	+0.4	+0.9	−0.7	−1.0	−3.7	-0.8 ± 1.8
iTransformer (prof)	PEMS04	−0.9	+1.5	+0.8	−0.7	+0.5	$+0.2 \pm 1.0$
PatchTST	ECL	−4.8	−5.4	−5.8	−4.6	−3.6	-4.8 ± 0.8
DLinear	ECL	−3.3	−2.6	−3.1	−3.2	−3.6	-3.2 ± 0.4
PaiFilter	ECL	−1.0	−1.1	−0.7	−1.0	−0.8	-0.9 ± 0.2
iTransformer	ECL	+1.8	+2.8	+3.9	+2.0	+1.5	$+2.4 \pm 1.0$
PatchTST	Solar	+6.1	+2.3	+0.6	+5.5	+2.4	$+3.4 \pm 2.3$
DLinear	Solar	−2.3	−1.5	−2.6	−3.8	−2.6	-2.6 ± 0.8
PaiFilter	Solar	−1.8	−3.2	−4.7	−3.7	−2.7	-3.2 ± 1.1
iTransformer	Solar	−6.9	−0.1	+4.1	+2.1	+6.1	$+1.1 \pm 5.0$

Seed profiles separate channel correlation from the opportunity left by a predictor. Solar has the highest $\bar{r}$, yet its CI gains are closer to ECL than to PEMS04; the normalized residual gaps in Table A.10 order backbones within PEMS04.

Forecast horizon introduces another source of variation in complete-system performance. PEMS08's MSE reduction decreases from 48.9% at h=12 to 37.1% at h=96, while PEMS04/07 have larger reductions at longer horizons (Table A.14). The headroom analysis compares fixed-horizon, fixed-backbone pairs, separating this ordering from the horizon dependence of complete-system gains.

ETTm1 and ETTm2 further distinguish static control from dynamic state conditioning in a direct-mixing backbone. Their channel correlations are $\bar{r}$=0.314/0.349. Against same-recipe five-seed no-instruction references, static controls change MSE by -1.4%/-0.4% and dynamic instructions by -0.3%/$+0.2\%$. ETTm1 static control improves on four of five reference seeds and beats dynamic control on all five, averaging 1.1%. Reference MSEs are 0.3331 ± 0.0027/0.1807 ± 0.0010, static MSEs are 0.3284 ± 0.0041/0.1799 ± 0.0011, and dynamic MSEs are 0.3320 ± 0.0038/0.1809 ± 0.0015. Here the observed benefit comes chiefly from static control, while dynamic state conditioning contributes the smaller change.

H COMPILER REUSE AND MATCHED INPUT-SCOPE EXPERIMENTS

The interface's central reuse question is whether a compiler learned with one predictor can supply useful state to another. A second question concerns information access: how much of a package's benefit comes from local adaptation, and how much comes from shared observations? This section gives the protocols and full results for both comparisons.

Input and forecast lengths are 96, message width is 128, and seeds are 2021–2025. Base denotes the backbone alone. Local compiles the target channel's history; Cross-channel compiles the channel-averaged window. Both combine two profile and two static vectors at identical parameter counts within each backbone. The training objective is forecast MSE, so the scope comparison measures the value of the supplied observations directly.

The scope comparison covers six backbones on PEMS04 and DLinear on ECL, with separately trained Base, Local, and Cross-channel models. The FITS and PEMS04 DLinear configurations are those in Table A.6. PatchTST uses batch 16; both input-scope models have 949,857 parameters.

PatchTST uses four layers, 16 heads, FFN width 512, learning rate 0.001, at most 80 epochs, patience 15, and cosine scheduling. PaiFilter and TexFilter use batch 16, hidden/embedding width 256, learning rate 0.005, at most 20 epochs, patience 6, and type1 scheduling. DLinear and C-LoRA RMLP use batch 32, learning rate 0.0003, at most 50 epochs, patience 5, and cosine scheduling. C-LoRA uses rank 16 and node dimension 32 from the official RMLP implementation (Nie et al., 2024), with model width 128 in this forecasting pipeline. Adam and dropout 0.1 are used throughout, and the checkpoint with lowest validation loss is evaluated on the test split.

H.1 Frozen compiler reuse

Freezing the source compiler separates the value of its learned expression from the target reader's adaptation. The fixed transfer matrix covers five settings, three conditions, and seeds 2021–2025, totaling 75 target runs. A matched ten-run comparison evaluates the token-preserving, horizon-aware residual reader in Eq. 2 for PatchTST→DLinear; it compares frozen learned and frozen random compilers over the same five seeds. Each source is the corresponding seed's PEMS04 multichannel checkpoint: the forecast-MSE-trained FITS, DLinear, or PatchTST cross-channel model specified above. We transfer the profile compiler and static vectors, leaving the target backbone and reader newly initialized.

The frozen learned and frozen random conditions match architecture, frozen/trainable parameter counts, target initialization, data order, preprocessing, and optimization. Joint training from scratch uses the same target initialization and random compiler as the random condition, but optimizes all parameters. This separates source reuse from learning a new target interface.

Preserving the compiler function. The transferred package has two static and two profile vectors of width 128. FITS and DLinear average dataset-standardized channels at each observed time step. PatchTST first centers each channel within the window and divides by the square root of population variance plus 10^{-5}, then averages channels. All three conditions in the PatchTST-source setting preserve this preprocessing. The two frozen conditions hold all compiler weights fixed, including the unused auxiliary head. Compiler dropout remains active during target training and is disabled at evaluation. The transfer objective is forecast MSE.

PEMS04 and PEMS08 have 307 and 170 channels at five-minute intervals. Both use 96 observed and 96 predicted steps. PEMS08 retains its own training-fitted scaler and chronological splits. The frozen compiler therefore generates new messages from target histories, while the new backbone and residual reader learn from target labels. The transferred object is the learned message-generating function. The FITS compiler supplies reusable guidance to DLinear on both datasets, reducing MSE by 11.32% on PEMS04 and 12.66% on PEMS08 relative to frozen-random compilers. The targeted PatchTST→DLinear comparison uses the ordered four-token message and horizon-aware residual reader; it gives a 7.56% reduction with all five seeds improving. The separate input-scope comparison measures the value of shared observations (Table A.22). The learned and random token-horizon arms each train 96,355 target parameters; the frozen-random contrast remains parameter- and initialization-matched.

Target training and estimation. FITS and DLinear use batch 32, learning rate 0.0003, at most 50 epochs, patience 5, and cosine scheduling. PaiFilter uses batch 16, learning rate 0.005, at most 20 epochs, patience 6, type1 scheduling, hidden width 256, and its native length-96 embedding. All use Adam, forecast MSE, dropout 0.1, full precision, and no augmentation. Validation selects the checkpoint before evaluation on the full test split.

Table A.18 reports every seed and condition. Reuse gain is $100(1 - \bar{e}_{\text{learned}}/\bar{e}_{\text{random}})$, with means over five seeds. The paired-seed bootstrap 95% intervals in Table 2(b) use 20,000 resamples (seed 1077) for *absolute* MSE reduction. They quantify training-seed variation on the fixed datasets and recipes.

Table A.18: Seed-level MSE for frozen-compiler reuse. All sources are trained on PEMS04, and group headings identify target data. Reduction compares frozen learned and frozen random compilers with the same target. † denotes the token-preserving, horizon-aware residual reader.

Seed	Frozen random	Frozen learned	Joint scratch	Reduction ↑
FITS → DLinear / PEMS04				
2021	0.38161	0.33713	0.34543	11.66%
2022	0.38338	0.33805	0.34070	11.82%
2023	0.38092	0.33835	0.34224	11.18%
2024	0.37951	0.33979	0.34345	10.47%
2025	0.38258	0.33863	0.34449	11.49%
DLinear → FITS / PEMS04				
2021	0.57410	0.51852	0.51411	9.68%
2022	0.56685	0.51801	0.50655	8.62%
2023	0.57043	0.51845	0.51006	9.11%
2024	0.56970	0.51813	0.50623	9.05%
2025	0.56738	0.52122	0.50476	8.13%
FITS → PaiFilter / PEMS04				
2021	0.32955	0.31738	0.31572	3.69%
2022	0.33664	0.32421	0.31600	3.69%
2023	0.32971	0.31696	0.31492	3.87%
2024	0.32721	0.31772	0.31353	2.90%
2025	0.32291	0.31486	0.31235	2.49%
PatchTST → DLinear / PEMS04 (pooled scalar reader)				
2021	0.37985	0.37883	0.32540	0.27%
2022	0.37508	0.37454	0.32527	0.14%
2023	0.37553	0.37519	0.33212	0.09%
2024	0.37651	0.37534	0.32559	0.31%
2025	0.37846	0.38003	0.32736	-0.42%
PatchTST → DLinear† / PEMS04				
2021	0.29601	0.28106	–	5.05%
2022	0.30216	0.27710	–	8.29%
2023	0.30013	0.27205	–	9.36%
2024	0.29504	0.27297	–	7.46%
2025	0.30159	0.27868	–	7.59%
FITS → DLinear / PEMS08				
2021	0.51451	0.45077	0.46339	12.39%
2022	0.51751	0.45038	0.45946	12.97%
2023	0.51492	0.45243	0.46276	12.14%
2024	0.51738	0.45079	0.46401	12.87%
2025	0.51936	0.45225	0.46300	12.92%

Table A.19: Five-seed compiler-reuse MAE (mean±sample SD) for the same frozen-random, frozen-learned, and joint-scratch conditions as the MSE comparison; the token-horizon row reports the frozen learned/random comparison.

Source → target / data	Frozen random	Frozen learned	Joint scratch
FITS → DLinear / PEMS04	0.46137 ± 0.00128	0.43348 ± 0.00063	0.43517 ± 0.00168
DLinear → FITS / PEMS04	0.59614 ± 0.00209	0.56602 ± 0.00111	0.56328 ± 0.00320
FITS → PaiFilter / PEMS04	0.42599 ± 0.00448	0.42085 ± 0.00319	0.41680 ± 0.00138
PatchTST → DLinear / PEMS04	0.45093 ± 0.00229	0.45334 ± 0.00118	0.42972 ± 0.00340
PatchTST → DLinear† / PEMS04	0.39791 ± 0.00359	0.38042 ± 0.00263	$--$
FITS → DLinear / PEMS08	0.50978 ± 0.00124	0.47456 ± 0.00045	0.47365 ± 0.00159

Source learning remains useful after replacing the predictor in the fixed matrix: FITS-to-DLinear, DLinear-to-FITS, FITS-to-PaiFilter, and cross-dataset FITS-to-DLinear reduce MSE in every seed, with positive paired intervals. Their mean MAE also improves over frozen-random compilers (Table A.19). The targeted cross-operation comparison adds a different compatibility result: preserving the ordered PatchTST compiler message tokens and decoding them into a horizon-dependent residual reduces MSE by 7.56% with 5/5 improving seeds and interval $[0.01856, 0.02602]$. The pooled-scalar control is retained in the seed table as the state-compressing comparison; the reader's operation determines which part of the learned expression is available to the target.

On PEMS08, the frozen learned compiler also lowers MSE by 2.42% relative to joint training from scratch (0.45133 versus 0.46252). The corresponding MAEs are 0.47456 for frozen reuse and 0.47365 for joint training (Table A.19).

Source-fitted geometry and target-reader response. The cross-dataset result raises a representational question: does the target reader use directions already organized by the source compiler? For each FITS-to-PEMS08-DLinear seed, we extract the 128-dimensional mean of the two static and two dynamic vectors, exactly as consumed by the residual reader. Dataset-specific training scalers and channel averaging stay unchanged. Windows have length 96 and stride 24. PCA is fitted to PEMS04 training messages; its center $\bar{\mathbf{m}}_S$ and two orthonormal directions $U \in \mathbb{R}^{128\times 2}$ are then fixed on both test sets. Figure A.2 applies these source coordinates directly to seed 2021. Color denotes the least-squares slope of the observed aggregate profile along a time axis spanning $[-1, 1]$.

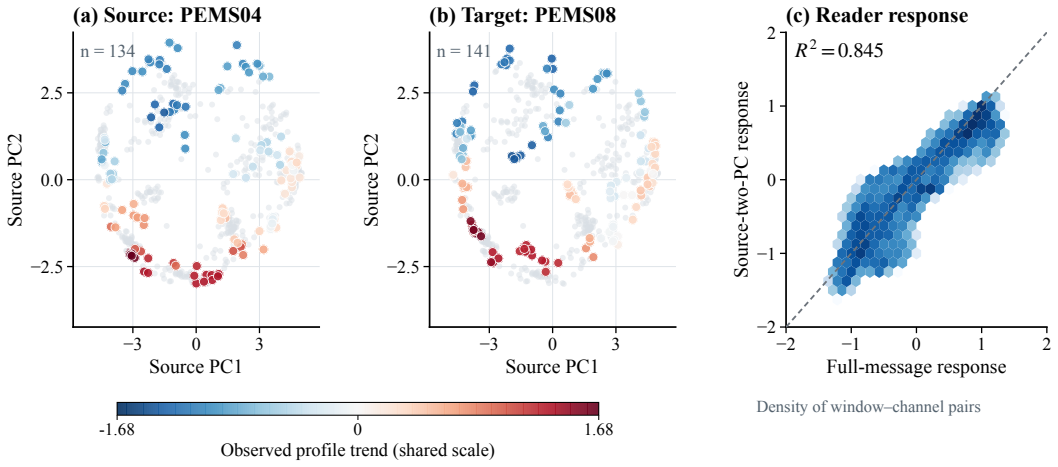


Figure A.2: **Source message geometry remains useful to a target residual reader on a new dataset.** FITS-to-DLinear reuse, seed 2021. (a,b) Source and target test messages in the same source-fitted PCA coordinates; gray points are source training messages. Color shows aggregate-profile slope on a shared scale. (c) Responses to full and source-two-PC messages under the same target reader. Hexagons count window–channel pairs; the dashed line is identity.

Variance capture measures how much target-message variation lies in the two source directions, dividing their variance by total target variance and centering each on its sample mean. To test the reader correction carried by these directions, reconstruct $\tilde{\mathbf{m}} = \bar{\mathbf{m}}_S + UU^\top(\mathbf{m} - \bar{\mathbf{m}}_S)$ and apply the unchanged target reader r_ϕ to both messages. Reader-response reconstruction is measured by

$$R^2_{\text{reader}} = 1 - \frac{\sum_{i,c}[r_\phi(\mathbf{m}_i)_c - r_\phi(\tilde{\mathbf{m}}_i)_c]^2}{\sum_{i,c}[r_\phi(\mathbf{m}_i)_c - \bar{r}_c]^2}, \qquad \bar{r}_c = \frac{1}{N}\sum_i r_\phi(\mathbf{m}_i)_c. \tag{8}$$

The source directions retain $91.96 \pm 0.25\%$ of source-test and $91.02 \pm 0.33\%$ of target-test message variance. Reader-response reconstruction gives $R^2 = 0.8471 \pm 0.0081$ (five-seed mean$\pm$sample SD). The source geometry captures much of the target expression and the reader's adjustment.

Source and target costs. We report source training and target fitting separately. Source training takes 122.46 ± 23.84 seconds per seed for FITS, 83.76 ± 11.77 for DLinear, and 4330.05 ± 318.34

for PatchTST (mean±sample SD; source GPU assignments vary by model). Target runs share the A100 server environment. Table A.20 gives fitting-plus-evaluation times, excluding source training. A single FITS source serves three target settings without retraining.

Each frozen compiler contains 6,913 parameters. The trainable target counts are 58,227 for PEMS04 DLinear, 40,443 for PEMS04 FITS, 89,817 for PEMS04 PaiFilter, and 40,554 for PEMS08 DLinear. Joint training retains the same total size and additionally optimizes the 6,913 compiler parameters.

Table A.20: Target fitting plus final evaluation time in seconds (mean±sample SD, five seeds). Source training is excluded and reported separately above.

Source → target / data	Frozen random	Frozen learned	Joint scratch
FITS → DLinear / PEMS04	43.37 ± 15.65	43.99 ± 6.80	66.80 ± 30.72
DLinear → FITS / PEMS04	82.61 ± 12.37	77.93 ± 15.54	86.85 ± 24.71
FITS → PaiFilter / PEMS04	66.74 ± 11.54	66.91 ± 9.58	80.61 ± 14.70
PatchTST → DLinear / PEMS04	49.49 ± 4.28	40.66 ± 4.63	61.94 ± 18.06
FITS → DLinear / PEMS08	33.38 ± 4.02	32.51 ± 7.55	27.22 ± 5.04

H.2 Complete input-scope comparisons

The matched input-scope study asks a different question from frozen reuse: what information should a compiler receive during training? Table A.21 compares Base, Local, and Cross-channel conditions. C-LoRA RMLP processes histories independently with channel-specific low-rank parameters, so its Local versus Cross-channel contrast adds shared observations beyond parameter personalization. TexFilter already mixes channels in its base path; its contrast changes the added adapter's observations.

Table A.21: Forecast response to matched compiler input scope. MSE is five-seed mean±sample SD. Local and Cross-channel have the same parameter count. All conditions use forecast MSE and the training settings specified above.

Dataset / backbone	Base	Local	Cross-channel
PEMS04 / C-LoRA RMLP	0.41333 ± 0.00455	0.26543 ± 0.00240	0.26282 ± 0.00267
PEMS04 / PatchTST	0.44674 ± 0.00396	0.44495 ± 0.00599	0.26024 ± 0.00137
PEMS04 / PaiFilter	0.56838 ± 0.00442	0.31680 ± 0.00207	0.31444 ± 0.00158
PEMS04 / TexFilter	0.24187 ± 0.00131	0.17724 ± 0.00176	0.18122 ± 0.00121
PEMS04 / FITS	1.15622 ± 0.00208	0.52794 ± 0.00411	0.50727 ± 0.00157
PEMS04 / DLinear	0.42847 ± 0.00043	0.37396 ± 0.00098	0.34385 ± 0.00055
ECL / DLinear	0.19484 ± 0.00025	0.18916 ± 0.00075	0.18830 ± 0.00021

Let B, L, X denote the five-seed mean errors for Base, Local, and Cross-channel. Local adaptation, additional input scope, and the full package give reductions of $100(1 - L/B)$, $100(1 - X/L)$, and $100(1 - X/B)$, respectively. Table A.22 keeps these denominators explicit because the three reductions answer distinct sequential questions. Intervals use 20,000 paired resamples (seed 31077) to quantify training-seed variation on fixed data and recipes.

Table A.22: Relative MSE reductions and paired-seed bootstrap 95% intervals (%). Positive values indicate lower error after the indicated change.

Dataset / backbone	Base to Local	Local to Cross-channel	Base to Cross-channel
PEMS04 / C-LoRA RMLP	35.78 [34.98, 36.80]	0.99 [0.33, 1.57]	36.42 [35.88, 37.35]
PEMS04 / PatchTST	0.40 [−0.72, 2.01]	41.51 [40.80, 42.22]	41.75 [41.15, 42.26]
PEMS04 / PaiFilter	44.26 [43.96, 44.46]	0.75 [0.32, 1.10]	44.68 [44.44, 44.90]
PEMS04 / TexFilter	26.72 [26.11, 27.42]	−2.25 [−2.82, −1.80]	25.07 [24.45, 25.72]
PEMS04 / FITS	54.34 [54.03, 54.59]	3.92 [3.32, 4.74]	56.13 [56.02, 56.22]
PEMS04 / DLinear	12.72 [12.50, 12.84]	8.05 [7.82, 8.33]	19.75 [19.63, 19.84]
ECL / DLinear	2.92 [2.52, 3.28]	0.45 [0.19, 0.81]	3.35 [3.22, 3.47]

The additional value of shared observations depends on the reader. PatchTST, FITS, DLinear, and ECL DLinear have positive scope gains in all five seeds; C-LoRA RMLP and PaiFilter do so in four.

TexFilter has negative scope gains in every seed even though both Local and Cross-channel packages improve on its base. Separating local adaptation from input scope identifies which change supplies the package benefit for each reader.