LATENTFLOW: COMPONENTWISE NONLINEAR CONTINUATION FOR MULTIVARIATE TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multivariate forecasting architectures increasingly expose structure through decomposition, temporal scales, spectral routing, or variable-aware representations. Yet a distinct design question remains less directly isolated: once several structured temporal coordinates are available, *when should their coordinate axis be reduced relative to the nonlinear forecasting operator*? We study this representation–continuation interface with LATENTFLOW, an architecture for *componentwise nonlinear continuation*. A prefix-only encoder constructs structured coordinates, and shared coordinate-modulated variate–patch operators transform each coordinate separately through horizon projection and decoding; only the resulting future forecasts are then aggregated. A complementary linear bank handles simple extrapolative behavior. Under a controlled fixed-budget protocol with 2,048 matched training windows per task, LATENTFLOW achieves the lowest aggregate MSE and MAE among seven evaluated implementations, wins 23 of 28 seed-matched MSE comparisons against TimePro, and retains a 1.38% mean task-wise MSE advantage across five seeds. More importantly, a matched intervention that reduces the same coordinate axis *before* nonlinear continuation increases mean relative MSE by 5.12% (95% source–seed clustered interval $[2.42, 8.44]\%$). Conversely, replacing the GP-derived coordinates with a matched deterministic multibranch representation yields nearly identical aggregate point-forecasting error and an even 14–14 task split. Together, these results show that the main effect is not tied to a particular stochastic parameterization, but to the computation that consumes the representation: structured coordinates can be more effective when they remain computationally explicit through nonlinear continuation and are fused only after their futures are formed.

1 INTRODUCTION

Long-horizon multivariate forecasting increasingly relies on *structured* representations of the observed history. Decomposition methods expose trend and seasonal behavior, multiresolution models separate temporal scales, and variable-token or channel-routing architectures reorganize cross-series dependence. These approaches address an important question: *what structure should a forecaster represent?* We study a complementary question that has been less directly isolated: once several structured temporal coordinates are available, *where should that structure be reduced relative to the nonlinear forecasting operator?*

This distinction is consequential because aggregation and nonlinear continuation generally do not commute. Given coordinate states $Z_1, \ldots, Z_Q$, a forecaster may first reduce them to a single state and apply a nonlinear operator to the aggregate, or instead apply the operator to each coordinate and combine only the resulting future predictions. The two computations coincide for affine continuation but can differ substantially for nonlinear operators. Existing componentwise and decomposition-based forecasters establish that structured prediction is useful; our focus is narrower and orthogonal: the *representation–continuation interface*, specifically whether the coordinate axis should remain computationally explicit while a shared nonlinear multivariate operator acts on it.

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.

LATENTFLOW realizes this idea as *separate, continue, then fuse*. A prefix-only structural encoder first maps the observed history to a latent state and exposes recurring, smooth, and short-memory coordinates through a sparse additive representation. Each coordinate is then embedded into a variate–patch field and processed by the same local nonlinear continuation operator, with bounded coordinate-specific modulation. Crucially, the coordinate axis is retained through field evolution, horizon projection, and decoding; only the decoded future forecasts are fused. A complementary linear continuation bank captures simple extrapolative behavior. Algorithm 1 gives the end-to-end computation, Figure 1 maps its stages to the architecture, and Figure 2 isolates the single design variable studied in this paper: *the location of coordinate reduction*. Detailed construction and implementation are deferred to Sections 3.1–3.3 and Appendix A.

The central hypothesis is computational rather than tied to a particular stochastic representation. Lemma 1 shows that, when the fitted covariance operators are available, summing the additive posterior means need not make the individual coordinates algebraically unrecoverable. Proposition 1 instead identifies the relevant distinction: nonlinear continuation before and after coordinate reduction are generally different computations. LATENTFLOW therefore does not rely on an information-loss argument. It explicitly exposes coordinate-specific states to the nonlinear forecasting operator and tests whether retaining that axis through continuation provides a useful architectural inductive bias.

The experiments support this distinction. Under a controlled fixed-budget protocol with 2,048 matched training windows per task, LATENTFLOW achieves the lowest aggregate MSE and MAE among seven evaluated implementations and retains a 1.38% mean task-wise MSE advantage over TimePro across five seeds. More importantly, across 140 task–seed cells, moving coordinate reduction *before* nonlinear continuation increases mean relative MSE by 5.12% (95% source–seed clustered interval $[2.42, 8.44]\%$). In contrast, replacing the GP-derived coordinates with a matched deterministic multibranch representation produces nearly identical aggregate point-forecasting error and an even 14–14 split over task means. The benefit therefore does not depend on a uniquely stochastic frontend: the stronger and more general signal is that structured coordinates can be useful when the downstream nonlinear computation retains them explicitly until forecast formation. The fixed-budget comparison is used as a controlled operating point for this architectural question, rather than as a claim of universal full-data superiority.

Our contributions are:

- **A representation–continuation principle.** We formulate *componentwise nonlinear continuation*: preserve an explicit coordinate axis through nonlinear multivariate continuation, horizon projection, and decoding, and reduce that axis only in forecast space.
- **LATENTFLOW.** We instantiate this principle with a prefix-only structural encoder, sparse additive temporal coordinates, shared coordinate-modulated variate–patch fields, late forecast fusion, and a complementary linear continuation bank, without dense $C \times C$ stochastic covariance or explicit cross-coordinate exchange.
- **Controlled evidence for the interface.** A seven-model matched benchmark, a five-seed early-fusion intervention, a matched deterministic multibranch control, factorial analysis, and checkpoint-level audits isolate the effect of *where structured coordinates are reduced* from the particular coordinate parameterization, auxiliary priors, and model-selection path.

2 RELATED WORK

Temporal structure and decomposition. Informer shortens attention paths, Autoformer couples decomposition with autocorrelation, and FEDformer performs frequency-enhanced decomposition (Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022b). N-BEATS and N-HiTS forecast through basis expansion and hierarchical interpolation; ETSformer separates level, growth, and seasonality; Koopa evolves invariant and variant dynamics; and TimeMixer mixes decomposed components across temporal resolutions (Oreshkin et al., 2020; Challu et al., 2023; Woo et al., 2023; Liu et al., 2023; Wang et al., 2024). Pathformer routes examples through scale-specific pathways (Chen et al., 2024). These models motivate explicit temporal structure, but they do not isolate reduction placement inside a matched shared nonlinear multivariate continuation operator. LatentFlow does not claim decomposition or componentwise forecasting itself as new.

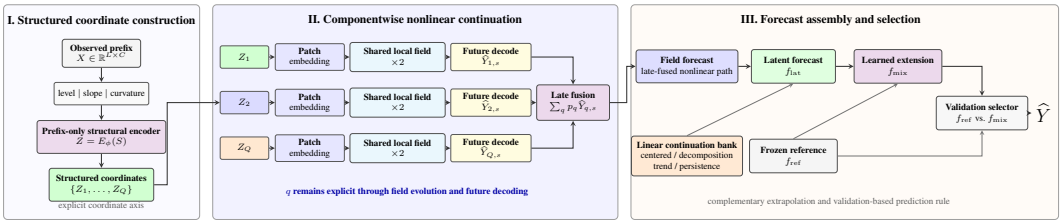

Figure 1: **LATENTFLOW.** Structured coordinates are continued separately through a shared nonlinear field and fused only after future decoding; complementary linear and reference paths complete the validation-selected forecast.

Patches, variables, and interaction operators. PatchTST represents each channel through temporal patches; SCINet recursively couples even and odd subsequences; TimesNet organizes variation by learned periods; and FiLM uses Legendre projections for long-range structure (Nie et al., 2023; Liu et al., 2022a; Wu et al., 2023; Zhou et al., 2022a). Crossformer models cross-time and cross-dimension dependence, while iTransformer treats variables as tokens (Zhang & Yan, 2023; Liu et al., 2024). The closest evaluated systems emphasize complementary interaction mechanisms: VPNet (2026) restricts cross-variate modeling to local variate–patch neighborhoods; xCPD (2026) routes channel–patch dependencies through graph-spectral decomposition; and TimePro (2025) constructs variable- and time-aware hyper-states (Liu et al., 2026; Li et al., 2026; Ma et al., 2025). DLinear (2023) supplies a strong decomposition-linear reference, and TimeMixer (2024) tests multiresolution mixing (Zeng et al., 2023; Wang et al., 2024). LatentFlow instead asks how long a structured coordinate axis should remain explicit while a shared nonlinear interaction operator acts on it.

Distribution shift and stochastic representations. RevIN and Non-stationary Transformers address distribution shift through reversible normalization and learned de-stationarization (Kim et al., 2022; Liu et al., 2022b). Gaussian processes and vector-valued kernels formalize temporal and multi-output covariance (Rasmussen & Williams, 2006; Roberts et al., 2013; Alvarez et al., 2012); inducing-variable objectives improve scalability (Snelson & Ghahramani, 2005; Titsias, 2009), while spectral kernels support pattern discovery (Wilson & Adams, 2013). DeepAR and Temporal Fusion Transformers learn predictive distributions (Salinas et al., 2020; Lim et al., 2021). Motion Code learns sparse multi-stochastic representations and informative timestamps (Bajaj & Nguyen, 2024). LatentFlow adopts sparse additive covariance coordinates as one structured representation for a neural point forecaster. The deterministic control shows that its central point-forecasting result should not be attributed to a uniquely stochastic advantage.

3 METHOD

Let $X \in \mathbb{R}^{L\times C}$ denote an observed prefix and $Y \in \mathbb{R}^{H\times C}$ its future. All representation, routing, and gating features are computed from the observed prefix only. Dataset-level standardization uses training-split statistics, followed by prefix-only per-channel normalization inside the model; we denote the resulting input by $\widetilde{X}$.

Method at three levels. Algorithm 1 gives the shortest operational view of LATENTFLOW, while Figure 1 maps the same stages to the architecture. Stage I constructs an explicit structured-coordinate axis (Section 3.1; Appendix A.1–A.2). Stage II is the central operation: a shared nonlinear variate–patch forecaster acts on each coordinate while that axis remains explicit, and reduction occurs only after horizon decoding (Section 3.2; Appendix A.3). Stage III assembles the nonlinear forecast with a complementary linear bank and performs validation-based predictor selection (Section 3.3; Appendix A.3–A.4). Stage IV states the architectural invariant isolated by Figure 2 and analyzed in Section 3.4 and Appendix B.

The name *LatentFlow* refers to the continuation of structured latent coordinates through the forecasting field; it is unrelated to normalizing flows, flow matching, or continuous-time flow models.

Algorithm 1 LATENTFLOW: componentwise nonlinear continuation

Require: Prefix $X \in \mathbb{R}^{L\times C}$, forecast horizon H
Ensure: Validation-selected forecast $f_{\mathrm{sel}}(X) \in \mathbb{R}^{H\times C}$

I. Construct structured coordinates
Section 3.1; Figure 1; Appendix A.1–A.2

1: Normalize the observed prefix and encode level, slope, and curvature features into a structural state Z.
2: Apply the sparse additive temporal representation to obtain explicit coordinates $\{Z_q\}_{q=1}^{Q}$ and their marginal summaries $\{V_q\}_{q=1}^{Q}$.
3: Fit the representation using the Gaussian, sparse-approximation, and signal-reconstruction terms in $\mathcal{L}_{\mathrm{rep}}$.

II. Continue each coordinate to the future
Section 3.2; Figures 1–2; Appendix A.3

4: **for** each coordinate q and resolution $s \in \mathcal{S}$ **do**
5: Patch-embed Z_q and apply the shared prototype-memory operator.
6: Apply two shared local variate–patch field blocks with bounded coordinate-specific modulation.
7: Project and decode a coordinate-specific future $\widehat{Y}_{q,s}$.
8: Fuse only the decoded futures, $\widehat{Y}_s = \sum_q p_q(X, H, V_q)\widehat{Y}_{q,s}$, and combine resolutions with the direct continuation path.

III. Assemble and select the forecast
Section 3.3; Figure 1; Appendix A.3–A.4

9: Reconstruct the signal and form centered, decomposition, trend, and persistence continuations; gate them into Y_{bank}.
10: Combine the componentwise nonlinear forecast with Y_{bank} to obtain the latent continuation f_{lat}.
11: Form the learned extension $f_{\mathrm{mix}}^{(e)} = \omega_0 f_{\mathrm{ref}} + \omega_1 f_{\mathrm{lat}}^{(e)}$.
12: Select the reference or an extension checkpoint using validation MSE: $e^* = \arg\min_{e\in\{\mathrm{ref},1:E\}} \mathrm{MSE}_{\mathrm{val}}(f_e)$.
13: **return** $f_{\mathrm{sel}} = f_{e^*}$.

IV. Core design invariant
Section 3.4; Figure 2; Appendix B
Keep the coordinate index q explicit through nonlinear field evolution, horizon projection, and decoding; reduce q only in forecast space.

3.1 STRUCTURED COORDINATE CONSTRUCTION

Stage I: construct an explicit coordinate axis. Following Stage I of Algorithm 1 and the left panel of Figure 1, we first map the observed prefix to a shared structural state. For each channel, the encoder exposes level, slope, and curvature information:

$$Z = E_\phi(S), \qquad S = [\widetilde{X}, \Delta\widetilde{X}, \Delta^2\widetilde{X}], \qquad Z \in \mathbb{R}^{L\times C\times D_z}. \tag{1}$$

The encoder is shared across channels and uses only the observed prefix. It combines a linear lift with a bounded left-padded convolutional residual, while a linear decoder D_ϕ reconstructs the observed signal.

We then expand Z into an explicit set of structured temporal coordinates. The construction uses trainable periodic, Matérn-$3/2$, and Ornstein–Uhlenbeck covariance families, which provide recurring, smooth, and short-memory biases. For coordinate q, learned ordered inducing timestamps U_q define the Nyström operator

$$Q_q = K_{TU_q}^{(q)}(K_{U_qU_q}^{(q)} + \epsilon I)^{-1}K_{U_qT}^{(q)}, \qquad C_g = \sum_q \pi_{gq}Q_q + \sigma_n^2 I + \epsilon I. \tag{2}$$

Here T is the observed temporal grid and π_{gq} is a simplex allocation over coordinates for sensor group g. Fixed semantic groups are used where channel identity is available, while high-dimensional anonymous datasets use prefix-conditioned soft groups. The resulting construction does not form a dense $C \times C$ covariance matrix.

For a temporal latent signal $z \in \mathbb{R}^L$, one Cholesky solve per group gives $\alpha_g = C_g^{-1}z$, from which the coordinate means are

$$m_{gq} = \pi_{gq}Q_q\alpha_g, \qquad z = \sum_q m_{gq} + (\sigma_n^2 + \epsilon)\alpha_g, \qquad (Z_q)_{b,:,c,d} = \sum_g r_{bcg}m_{gq,bcd}. \tag{3}$$

The tensors $\{Z_q\}_{q=1}^{Q}$ are the structured coordinates passed to Stage II. Their sum forms a denoised reconstruction of the structural state, while marginal second moments V_q are retained as routing

features. The point forecaster continues these coordinate means directly; no trajectory sampling is required.

The representation is trained with

$$\mathcal{L}_{\text{rep}} = \mathcal{L}_{\text{G}} + \sum_g \rho_g \frac{\sum_q \pi_{gq} \operatorname{tr}(K_q - Q_q)}{2\sigma_n^2 L} + 0.01\mathcal{L}_{\text{group}} + \mathcal{R}(D_\phi Z, \widetilde{X}) + \mathcal{R}(D_\phi \textstyle\sum_q Z_q, \widetilde{X}), \quad (4)$$

where ρ_g is mean group occupancy and $\mathcal{R}(A, B) = \|A - B\|_F^2/(\|B\|_F^2 + \epsilon)$. Kernel definitions, inducing-time parameterization, group assignments, posterior moments, and the full Gaussian term are given in Appendix A. The role of this stage is to expose a structured coordinate axis; the central architectural question is how the downstream forecasting operator consumes that axis.

3.2 Componentwise Nonlinear Continuation

Stage II: continue coordinates before reducing them. This stage implements the defining operation of LatentFlow (Algorithm 1; middle panel of Figure 1). Rather than first collapsing $\{Z_q\}$ into a single latent state, the same nonlinear forecasting operator acts on each coordinate while the coordinate index remains explicit.

For each patch resolution $s \in \mathcal{S}$, a shared adapter and patch encoder map Z_q to $H_{q,s}^0 \in \mathbb{R}^{D\times C\times P_s}$, where $P_s = \lceil L/s \rceil$. A prototype-memory operator acts along the patch axis, followed by two local variate–patch field blocks. Suppressing batch and scale indices, one block is

$$\Gamma^j(H) = H \odot [1 + 0.5\,\sigma(\text{MLP}(|\,\text{rFFT}(H)|))]\,, \quad (5)$$

$$F^j(H) = H + \text{FFN}^j\left(\text{GELU}\left[\text{BN}\left(\sum_{k\in\{1,3,7,17\}} a_k^j(H)\,\text{DWConv}_{k\times 3}(\Gamma^j(H))\right)\right]\right), \quad (6)$$

$$H_{q,s}^{j+1} = (1 + 0.1\tanh u_{q,s}^j) \odot F^j(H_{q,s}^j) + 0.1\tanh v_{q,s}^j, \qquad j \in \{0, 1\}. \quad (7)$$

The radius weights $a^j(H)$ are sample-conditioned simplex weights. The field operator, prototype memory, future projector, decoder, and normalization parameters are shared across coordinates at a fixed resolution; only the bounded FiLM parameters (u, v) provide coordinate-specific modulation. Thus distinct coordinates receive distinct nonlinear activations without requiring independently parameterized forecasting networks or explicit cross-coordinate exchange.

Crucially, future projection and decoding occur *before* any reduction over q:

$$\widehat{Y}_{q,s} = D_s(P_s^F H_{q,s}^2), \qquad \widehat{Y}_s = \sum_q p_q(X, H, V_q)\widehat{Y}_{q,s}. \quad (8)$$

The gate p_q is sample-conditioned using coordinate energy, marginal variance, and forecast horizon. Equation 8 is the *first reduction over the coordinate axis in the nonlinear path*. Therefore, q remains explicit through nonlinear field evolution, horizon projection, and future decoding, and only the resulting forecasts are fused.

This placement defines *componentwise nonlinear continuation*. Figure 2 isolates the same design choice by moving the reduction operation before the nonlinear operator while keeping the remaining continuation architecture matched. Implementation details for the field, gates, sharing scheme, and training are given in Appendix A.

3.3 Forecast Assembly and Validation Selection

Stage III: combine complementary continuation paths. Following Stage III of Algorithm 1 and the right panel of Figure 1, the componentwise nonlinear forecast is complemented by a lightweight bank for simple extrapolative structure. Let $\bar{X} = D_\phi(\sum_q Z_q)$ denote the reconstructed normalized signal. Four per-channel experts compute

$$Y_{\text{ctr}} = A_{\text{ctr}}(\bar{X} - \bar{X}_L) + \bar{X}_L, \qquad Y_{\text{dec}} = A_{\text{tr}}M(\bar{X}) + A_{\text{rem}}(\bar{X} - M(\bar{X})),$$
$$Y_{\text{trend}} = \text{OLSExtrapolate}(\bar{X}), \qquad Y_{\text{last}} = \mathbf{1}_H \bar{X}_L, \qquad Y_{\text{bank}} = \sum_b \beta_b(X) Y_b. \quad (9)$$

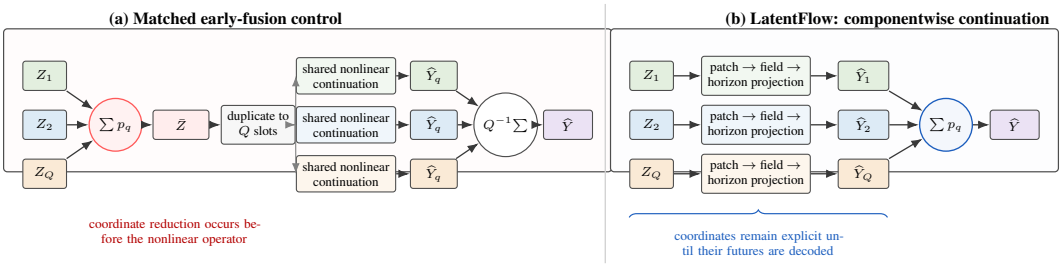

Figure 2: **Representation–continuation intervention.** The matched control reduces the same coordinates before nonlinear continuation, whereas LATENTFLOW preserves the coordinate axis through horizon decoding and fuses only the predicted futures.

Here M combines moving averages of widths 7 and 25, and the sample-conditioned gate $\beta(X)$ selects among centered, decomposition, trend, and persistence continuations.

The nonlinear path combines the componentwise forecasts from Stage II with a direct residual projection:

$$Y_{\text{field}} = Y_{\text{trend}} + \sigma(\kappa)\left(\gamma_0 Y_{\text{dir}} + \sum_{s\in\mathcal{S}} \gamma_s \widehat{Y}_s\right), \qquad f_{\text{lat}}(X) = \eta_0 Y_{\text{field}} + \eta_1 Y_{\text{bank}}. \tag{10}$$

The gates γ and η are sample-conditioned simplexes and are shared across channels and forecast steps. Thus the structured nonlinear continuation remains the primary forecasting path, while the bank supplies complementary low-complexity extrapolation.

After reversing prefix normalization, the resulting latent forecast is combined with a frozen deterministic reference:

$$f^{(e)}_{\text{mix}}(X) = \omega_0(X) f_{\text{ref}}(X) + \omega_1(X) f^{(e)}_{\text{lat}}(X), \tag{11}$$

where e indexes extension checkpoints. Extension training minimizes

$$\mathcal{L} = \text{MSE}(Y, f^{(e)}_{\text{mix}}(X)) + 0.02\,\mathcal{L}_{\text{NLL}} + \lambda_{\text{rep}}\mathcal{L}_{\text{rep}}. \tag{12}$$

The likelihood term is auxiliary to point forecasting; its specification is given in Appendix A.

The reported predictor is selected using validation MSE only:

$$e^* = \arg\min_{e\in\{\text{ref},1,\ldots,E\}} \text{MSE}_{\text{val}}(f_e), \qquad f_{\text{sel}} = \begin{cases} f_{\text{ref}}, & e^* = \text{ref}, \\ f^{(e^*)}_{\text{mix}}, & \text{otherwise}. \end{cases} \tag{13}$$

No test target enters this selection. All reported forecasting metrics use f_{sel}. Together, Stages I–III implement the complete architecture shown in Figure 1; the next subsection isolates the design choice that distinguishes LATENTFLOW from early coordinate reduction.

3.4 WHY THE TIMING OF FUSION MATTERS

The core question is not whether the coordinate tuple is recoverable in principle, but whether reducing it *before* a nonlinear forecasting operator induces the same computation as reducing the resulting forecasts afterward. Figure 2 isolates exactly this distinction.

Lemma 1 (Recovery of additive posterior means)**.** *Let $A_q \succeq 0$, $S = \sum_q A_q$, $C = S + \nu I$ with $\nu > 0$, and $m_q = A_q C^{-1} z$. If $u = \sum_q m_q$, then*

$$m_q = A_q S^{\dagger} u \tag{14}$$

for every q, including when S is rank deficient.

Lemma 1 shows that summing the additive posterior means does not necessarily erase them algebraically when the fitted covariance operators remain available. The motivation for componentwise continuation is therefore computational rather than an information-loss claim.

Proposition 1 (Nonlinear continuation and averaging). *Let p be a simplex, $\bar{z} = \sum_q p_q z_q$, and let F be twice differentiable on the convex hull of $\{z_q\}$ with Hessian operator norm at most M. Then*

$$\left\| \sum_q p_q F(z_q) - F(\bar{z}) \right\| \leq \frac{M}{2} \sum_q p_q \|z_q - \bar{z}\|^2. \tag{15}$$

The two computations coincide for affine F, but need not coincide for nonlinear F; for scalar $F(z) = z^2$, their difference is exactly the weighted variance.

Proposition 1 identifies the source of this discrepancy: nonlinear curvature interacting with dispersion across coordinates. It does not assert that one ordering must always produce lower forecast error; rather, it shows that

$$\sum_q p_q F(Z_q) \qquad \text{and} \qquad F\left(\sum_q p_q Z_q\right)$$

represent genuinely different computations once F is nonlinear.

Our matched intervention tests precisely this architectural choice. The early-fusion control forms $\bar{Z} = \sum_q p_q Z_q$ before the field operator, duplicates $\bar{Z}$ across the same Q slots, and otherwise retains the continuation depth, widths, optimization schedule, and validation-selection rule. LATENTFLOW instead keeps q explicit through nonlinear field evolution, horizon projection, and decoding, and performs the first reduction only in forecast space. Figure 2 therefore compares the two sides of the representation–continuation interface while holding the surrounding continuation architecture matched.

Theorem 1 and complete proofs are given in Appendix B.

4 EVALUATION

Controlled fixed-budget protocol. We evaluate ETTh1, ETTh2, ETTm1, ETTm2, Weather, Electricity, and Traffic with input length $L = 96$ and forecast horizons $H \in \{96, 192, 336, 720\}$, giving 28 dataset–horizon tasks. ETT follows the standard 12/4/4-month chronological partitions; the remaining datasets use chronological 70/10/20% splits. For every task, all compared methods receive the same 2,048 sampled training windows and the same 1,024 validation windows, with window indices fixed across methods and initialization seeds. Evaluation uses every valid test window. Dataset standardization is fitted on the training split only.

Why this regime matters. We use a bounded-supervision operating point to study architectural inductive bias under limited task-specific training. In this setting, explicitly retaining structured coordinates gives the continuation operator direct access to coordinate-specific computation rather than requiring such transformations to be recovered implicitly from an aggregated state. This regime is representative of settings such as task adaptation, periodic model refresh, and retraining under bounded data or compute. Our comparative claims are therefore made at this controlled operating point.

Compared methods. We compare LATENTFLOW with TimePro (2025), VPNet (2026), xCPD (2026), iTransformer (2024), TimeMixer (2024), and DLinear (2023). All results are produced by matched reruns rather than copied published scores. xCPD is evaluated as its graph-spectral routing module coupled to a DLinear host. Model implementations, optimization schedules, and final settings are specified in Appendix D.

Training and model selection. Each method uses its model-specific optimization schedule and the same sampled validation information for checkpoint selection. For LATENTFLOW, AdamW uses learning rate 7×10^{-4} and weight decay 10^{-4} (Loshchilov & Hutter, 2019); the deterministic reference is fitted first and then frozen while the continuation extension is trained. Dataset-level LATENTFLOW settings are selected using validation MSE and fixed across seeds. The reference is included as a valid selection candidate, as defined in Equation 13. No test target is used for gradient optimization, hyperparameter selection, or checkpoint selection. Complete training schedules and final configurations are given in Appendix D.

Table 1: Matched limited-window forecasting: 2,048 training windows, input length 96, seed 42 for every method. Dataset entries average four horizons; Overall averages all 28 tasks. Lower is better; bold and blue underline indicate best and second best.

Dataset	Metric	LatentFlow (Ours)	TimePro (2025)	VPNet (2026)	xCPD (2026)	iTransformer (2024)	TimeMixer (2024)	DLinear (2023)
ETTh1	MSE	0.4580	0.4614	0.4877	0.4657	**0.4545**	0.4707	0.4672
	MAE	**0.4463**	0.4499	0.4700	0.4606	0.4489	0.4538	0.4626
ETTh2	MSE	**0.3732**	0.3807	0.4042	0.5597	0.3817	0.3812	0.5495
	MAE	**0.3996**	0.4053	0.4193	0.5167	0.4062	0.4047	0.5175
ETTm1	MSE	0.4168	**0.4114**	0.4487	0.4127	0.4224	0.4207	0.4232
	MAE	0.4191	**0.4130**	0.4404	0.4145	0.4161	0.4148	0.4223
ETTm2	MSE	**0.2841**	0.2926	0.3055	0.3611	0.2960	0.2931	0.3706
	MAE	**0.3279**	0.3348	0.3469	0.4049	0.3388	0.3360	0.4157
Weather	MSE	**0.2511**	0.2642	0.2561	0.2646	0.2812	0.2763	0.2706
	MAE	**0.2794**	0.2852	0.2828	0.3140	0.2981	0.2974	0.3252
Electricity	MSE	**0.1823**	0.1868	0.1996	0.2084	0.2152	0.2234	0.2142
	MAE	**0.2748**	0.2758	0.2901	0.2983	0.2993	0.3081	0.3045
Traffic	MSE	**0.4585**	0.4615	0.4956	0.5531	0.5334	0.6220	0.6340
	MAE	**0.3022**	0.3027	0.3314	0.3406	0.3617	0.3957	0.3933
Overall	MSE	**0.3463**	0.3512	0.3711	0.4036	0.3692	0.3839	0.4185
	MAE	**0.3499**	0.3524	0.3687	0.3928	0.3670	0.3729	0.4059

Repetitions and statistical analysis. The seven-model matched comparison uses seed 42 for every method. We additionally repeat the strongest aggregate comparator, TimePro, together with LATENTFLOW over seeds 42–46, yielding 140 paired dataset–horizon–seed comparisons. Mechanism interventions use the corresponding repeated protocols specified with each experiment. We report arithmetic task averages, paired relative MSE changes, and task-wise wins separately. Confidence intervals use 20,000 clustered bootstrap resamples. The primary analysis treats five underlying sources as independent families: ETT station 1, ETT station 2, Weather, Electricity, and Traffic, while retaining all associated cadences, horizons, and seeds within each resampled family. We additionally report a source–seed clustered analysis for the five-seed comparisons. Appendix E.1 gives the complete estimands and statistical definitions.

5 RESULTS

5.1 MATCHED FIXED-BUDGET FORECASTING

Table 1 compares all seven implementations under identical sampled training windows, validation information, test windows, and seed. LATENTFLOW obtains the lowest aggregate MSE/MAE, $0.3463/0.3499$, with average ranks $1.64/1.75$, and the best MSE on 19 of 28 tasks. Against the strongest aggregate comparator, TimePro, it wins 23 of 28 tasks and reduces mean task-wise MSE by 1.87% (95% source-family interval $[0.35, 3.72]\%$).

The advantage persists across seeds 42–46. Over 140 paired dataset–horizon–seed comparisons, LATENTFLOW obtains MSE/MAE $0.34796/0.35079$ versus $0.35136/0.35243$ for TimePro and wins 94 MSE comparisons. Mean paired MSE improves by 1.38%, with 95% intervals $[0.14, 3.37]\%$ under source-family resampling and $[0.74, 2.14]\%$ under source–seed clustering. The gain remains positive when any one source family is omitted. Full five-seed results are reported in Appendix Table 10.

5.2 WHAT DOES COMPONENTWISE CONTINUATION CONTRIBUTE?

The matched early-fusion intervention provides the central mechanism test (Figure 2). Both variants use the same structured coordinates, continuation depth, widths, training schedule, and validation

Table 2: Five-seed mechanism interventions over all 28 tasks. Positive change means that the control is worse than the final componentwise-continuation model. Confidence intervals resample five source families with every horizon and seed retained. B/T/W denotes control better / tie / LatentFlow better.

Representation and continuation	MSE	MAE	Mean Δ MSE (%)	95% CI	B/T/W
LatentFlow	**0.3480**	**0.3508**	0.00	–	–
Causal $X \to Z$ without process separation	0.3635	0.3680	4.86	[0.88, 11.76]	22/29/89
Stochastic processes, fused before continuation	0.3623	0.3665	5.12	[0.63, 13.50]	33/25/82
Process-preserving continuation without linear bank	0.3498	0.3528	0.63	[0.04, 1.13]	49/0/91

rule; they differ only in whether the coordinate axis is reduced before or after nonlinear continuation. Across 28 tasks and five seeds, early fusion increases mean paired MSE by 5.12%, with 95% intervals $[0.63, 13.50]\%$ under source-family resampling and $[2.42, 8.44]\%$ under source–seed clustering. Replacing the separated coordinates entirely with the forecast-origin $X \to Z$ state similarly increases MSE by 4.86% ($[0.88, 11.76]\%$). The effect is especially large on Electricity ($+22.89\%$) and Traffic ($+8.96\%$), but it is not tied to the stochastic coordinate construction. A matched deterministic multibranch representation obtains MSE/MAE $0.34739/0.35037$ versus $0.34796/0.35079$ for the GP-derived coordinates, with an even 14–14 split across the 28 task means. Thus the final point-forecasting accuracy is not specific to the stochastic coordinate parameterization; both successful variants preserve an explicit coordinate axis through nonlinear continuation. A balanced $2 \times 2 \times 2$ factorial experiment independently supports this conclusion. Marginalizing over resolution and linear-bank state, componentwise continuation changes MSE by -0.01294 with dataset-clustered 95% interval $[-0.02676, -0.00043]$. The linear bank has a smaller complementary effect, while resolution shows no aggregate main effect.

Validation-selection audit. The gain is not explained by repeated fallback to the frozen reference. Across 140 five-seed fits, validation selects the learned extension in 108 cases (77.1%), and it is the lower-MSE test candidate in 98. The selected predictor obtains MSE 0.34796 versus 0.35138 for always retaining the reference, a 1.22% mean paired improvement ($[0.22, 3.01]\%$). Full checkpoint audits and coordinate-prior perturbations are reported in Appendix F.

6 Discussion and Limitations

The central result concerns the *representation–continuation interface*: within the matched architecture, reducing structured coordinates before nonlinear continuation is worse than retaining them through field evolution, horizon projection, and decoding. This is a computational rather than information-theoretic effect: Lemma 1 shows that additive components can remain algebraically recoverable after summation, whereas Proposition 1 shows that nonlinear continuation and averaging generally do not commute. The early-fusion intervention and factorial analysis support this distinction, with the largest effects on Electricity and Traffic.

The deterministic multibranch control further shows that the effect is not specific to GP-derived coordinates or particular covariance families: learned temporal branches achieve essentially the same aggregate forecasting accuracy while preserving the benefit of componentwise continuation. We therefore interpret the covariance construction as one structured way to instantiate recurring, smooth, and short-memory coordinates, not as uniquely identified physical processes. Our comparative claims are scoped to the controlled 2,048-window regime; the architecture also incurs additional cost from multiple coordinate paths and depends on channel ordering. Computational scaling and ordering sensitivity are reported in Appendices J and I.

7 Conclusion

LatentFlow studies when structured temporal coordinates should be fused relative to nonlinear multivariate forecasting. Preserving the coordinate axis through nonlinear continuation and fusing only predicted futures outperforms matched early fusion, while the deterministic multibranch control shows that final point-forecasting accuracy is not tied to GP-derived coordinates.

Reproducibility Statement

Appendices A and C specify tensor shapes, boundary conventions, objectives, splits, training schedules, selected hyperparameters, and the validation selector. Appendix B provides the proofs, and the remaining appendices give per-horizon results and audits. The complete repository will include model and baseline implementations, immutable experiment configurations, sampled window indices, evaluation and statistics code, result manifests, and checkpoint hashes.

AI Use Statement

Generative AI tools were used for coding, literature search and paper drafting assistance. Reported benchmark measurements come from executed experiments rather than generated observations. The authors retain responsibility for the scientific claims, implementation, and presentation.

Ethics Statement

This work uses public forecasting benchmarks and does not collect new personal data. Improved forecasting does not establish safety for operational decisions in infrastructure or environmental systems. Deployment would require domain-specific validation, monitoring, and consideration of the consequences of forecasting errors.

References

Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266, 2012. doi: 10.1561/2200000036.

Chandrajit Bajaj and Minh Nguyen. Motion code: Robust time series classification and forecasting via sparse variational multi-stochastic processes learning. *arXiv preprint arXiv:2402.14081*, 2024. URL https://arxiv.org/abs/2402.14081.

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. N-HiTS: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6989–6997, 2023. doi: 10.1609/aaai.v37i6.25854.

Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=lJkOCMP2aW.

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.

Dongyuan Li, Shun Zheng, Chang Xu, Jiang Bian, and Renhe Jiang. Routing channel-patch dependencies in time series forecasting with graph spectral decomposition. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=uIPAuyno4Z.

Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021. doi: 10.1016/j.ijforecast.2021.03.012.

Kun Liu, Renjun Jia, Ruifeng Yang, Xirui Zeng, Yuqi Liang, and Cen Chen. Are global dependencies necessary? scalable time series forecasting via local cross-variate modeling. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=CNVL194fO5.

Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. SCINet: Time series modeling and forecasting with sample convolution and interaction. In *Advances in Neural Information Processing Systems*, volume 35, 2022a.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems*, volume 35, 2022b.

Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with koopman predictors. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0538.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Xiaowen Ma, Zhen-Liang Ni, Shuai Xiao, and Xinghao Chen. TimePro: Efficient multivariate long-term time series forecasting with variable- and time-aware hyper-state. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 42096–42111, 2025. URL https://proceedings.mlr.press/v267/ma25p.html.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neill Gibson, and Suzanne Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A*, 371(1984):20110550, 2013. doi: 10.1098/rsta.2011.0550.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191, 2020. doi: 10.1016/j.ijforecast.2019.07.001.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

Michalis K. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 567–574, 2009. URL https://proceedings.mlr.press/v5/titsias09a.html.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. TimeMixer: Decomposable multiscale mixing for time series forecasting. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7oLshfEIC2.

Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1067–1075, 2013. URL https://proceedings.mlr.press/v28/wilson13.html.

Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. ETSformer: Exponential smoothing transformers for time-series forecasting. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=5m_3whfo483.

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*, volume 34, pp. 22419–22430, 2021.

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

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11121–11128, 2023. doi: 10.1609/aaai.v37i9.26317.

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.

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.

Tian Zhou, Ziqing Ma, Xue Wang, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, and Rong Jin. FiLM: Frequency improved legendre memory model for long-term time series forecasting. In *Advances in Neural Information Processing Systems*, volume 35, 2022a.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27268–27286, 2022b. URL https://proceedings.mlr.press/v162/zhou22g.html.

Table 3: Initial group–process probabilities. Entries follow the process inventory order. ETT and Weather memberships are fixed; behavioral-group priors are $(0.626168, 0.355140, 0.018692)$ for Electricity and $(0.406032, 0.402552, 0.191415)$ for Traffic.

Domain	Sensor group	Initial allocation
ETT	electrical load	0.48, 0.27, 0.05, 0.20
ETT	oil temperature	0.18, 0.07, 0.65, 0.10
Weather	pressure density	0.04, 0.30, 0.04, 0.03, 0.01, 0.58
Weather	thermodynamic	0.24, 0.46, 0.06, 0.03, 0.01, 0.20
Weather	humidity	0.14, 0.14, 0.48, 0.05, 0.04, 0.15
Weather	wind	0.08, 0.06, 0.04, 0.66, 0.08, 0.08
Weather	precipitation	0.05, 0.03, 0.10, 0.07, 0.70, 0.05
Weather	radiation	0.72, 0.10, 0.03, 0.03, 0.07, 0.05
Weather	atmospheric composition	0.18, 0.12, 0.08, 0.04, 0.03, 0.55
Electricity	strongly periodic	0.48, 0.30, 0.17, 0.05
Electricity	irregular high variability	0.17, 0.10, 0.13, 0.60
Electricity	low activity	0.15, 0.10, 0.60, 0.15
Traffic	strongly periodic	0.48, 0.30, 0.17, 0.05
Traffic	irregular high variability	0.17, 0.10, 0.13, 0.60
Traffic	low activity	0.15, 0.10, 0.60, 0.15

A Additional Method Details

A.1 Structural State and Sensor Allocation

Let L denote prefix length, H forecast length, C channel count, D_z structural width, D field width, Q coordinate count, M inducing count, and G sensor-group count. Batch size is B.

The structural encoder is

$$Z = W_0 S + \tfrac{1}{4}\sigma(a)\tanh\big[C_1\,\mathrm{GELU}\big(C_5^{\mathrm{causal}}(W_0 S)\big)\big]\,. \tag{16}$$

Here C_5^{causal} is a width-five left-padded temporal convolution with 32 hidden channels, and C_1 is pointwise. First and second differences are left-padded with zeros. At initialization, the linear lift places the signal level in its first coordinate, the residual projection is zero, and the decoder reads that coordinate, yielding an identity reconstruction. All encoder and decoder maps share parameters across observed channels.

ETT uses two fixed sensor groups: six load channels and oil temperature. Weather uses seven groups covering pressure/density, thermodynamic variables, humidity, wind, precipitation, radiation, and atmospheric composition. Electricity and Traffic instead use three prefix-conditioned behavioral groups derived from six summaries: mean absolute level, standard deviation, mean absolute first difference, maximum magnitude normalized by scale, and two lag correlations. The lags are $\min(\text{samples per day}, L/2)$ and half that value.

Group–coordinate allocation is

$$\pi_{gq} = \mathrm{softmax}_q\big[\log \pi^0_{gq} + 1.25\tanh b_{gq}\big]\,, \tag{17}$$

with $\sum_q \pi_{gq} = 1$. Learned behavioral memberships use the analogous simplex parameterization, $r_{bcg} > 0$ and $\sum_g r_{bcg} = 1$, whereas fixed semantic memberships are one-hot. The behavioral-group regularizer is the mean $\mathrm{KL}(r_{bc} \,\|\, r^0)$; it is zero for fixed groups. Initialization values are listed in Table 3.

Table 4: Temporal coordinate inventory and initialization. Periodic widths are dimensionless; remaining length scales are specified in days before conversion to dataset cadence.

Domain	Coordinate family	Initial period or length
ETT	daily, weekly / periodic smooth / Matérn; short-memory / OU	periods 1, 7; widths 0.65, 1.25 lengths 2, 0.20
Weather	daily / periodic smooth_1, smooth_2 / Matérn short_1, short_2 / OU slow / Matérn	period 1; width 0.65 lengths 2, 0.75 lengths 0.20, 0.06 length 7
Electricity	daily, weekly / periodic smooth / Matérn; short-memory / OU	periods 1, 7; widths 0.75, 1.25 lengths 7, 0.20
Traffic	daily, weekly / periodic smooth / Matérn; short-memory / OU	periods 1, 7; widths 0.75, 1.25 lengths 7, 0.15

A.2 Temporal Kernels and Inducing Timestamps

For $d = |t - t'|$, the temporal covariance families are

$$k_{\text{per}}(t, t') = a \exp[-2 \sin^2(\pi d/P)/\ell^2 - d/\delta], \tag{18}$$

$$k_{\text{Mat}}(t, t') = a(1 + \sqrt{3}d/\ell) \exp(-\sqrt{3}d/\ell), \tag{19}$$

$$k_{\text{OU}}(t, t') = a \exp(-d/\ell). \tag{20}$$

Positive parameters use softplus, while periodicities are parameterized as $P = P_0 \exp(0.2 \tanh b_P)$. Periodic length scales operate in phase units; the remaining scales are expressed in sampling steps after cadence conversion. Dataset-specific initial values are shown below.

Ordered inducing locations are parameterized through positive gaps $d_{qj} = \text{softplus}(\xi_{qj}) + 10^{-4}$:

$$u_{qm} = (L - 1) \frac{\sum_{j=1}^{m} d_{qj}}{\sum_{j=1}^{M+1} d_{qj}}, \qquad m = 1, \ldots, M. \tag{21}$$

They are initialized uniformly and remain ordered inside $(0, L - 1)$. Numerical jitter is fixed at $\epsilon = 10^{-4}$. All inverse expressions in the main text are implemented with Cholesky solves.

A.3 Gaussian Objective and Coordinate Moments

Writing $z_{bcd} = Z_{b,:,c,d}$, the normalized Gaussian term is

$$\mathcal{L}_{\text{G}} = \frac{1}{BCD_z} \sum_{b,c,d,g} r_{bcg} \frac{z_{bcd}^{\top} C_g^{-1} z_{bcd} + \log |C_g| + L \log(2\pi)}{2L}, \qquad \rho_g = \frac{1}{BC} \sum_{b,c} r_{bcg}. \tag{22}$$

The inducing correction used in Equation 4 is

$$\sum_g \rho_g \frac{\sum_q \pi_{gq} \operatorname{tr}(K_q - Q_q)}{2\sigma_n^2 L}, \tag{23}$$

following the standard sparse variational construction (Titsias, 2009).

With $A_{gq} = \pi_{gq} Q_q$, observed-time marginal variance is

$$v_{gq} = \text{diag}(\pi_{gq} K_q) - \text{diag}(A_{gq} C_g^{-1} A_{gq}). \tag{24}$$

For soft group assignments,

$$m_q = \sum_g r_g m_{gq}, \qquad v_q = \sum_g r_g (v_{gq} + m_{gq}^2) - m_q^2. \tag{25}$$

Table 5: Dataset-level hyperparameters selected by mean validation MSE across four horizons; test labels were not accessed.

Dataset	D_z	M	Patch lengths	λ_{rep}
ETTh1	16	32	16,32,48	0.05
ETTh2	8	32	16,32,48	0.001
ETTm1	8	32	8,24,48	0.01
ETTm2	8	16	8,16,32	0.001
Weather	16	32	8,16,32	0.001
Electricity	16	32	8,16,32	0.001
Traffic	16	32	8,16,32	0.001

Variances are lower-bounded by 10^{-8} for numerical stability. Future moments use inducing-based future–prefix covariances and the corresponding kernel diagonals. These second moments are supplied to the coordinate gate in Stage II.

Equation 3 follows directly from $C_g\alpha_g = z$. The reconstructed structural signal is obtained from $\sum_q Z_q$, while the reference and reconstruction objectives retain information complementary to the coordinate paths.

A.4 Fields, Gates, and Training

The continuation field uses width 128, two field blocks, and dropout 0.05. For patch scale s, a scale-specific adapter maps the sD_z structural features to s values before patch encoding. Local depthwise kernels use variable-axis widths $\{1, 3, 7, 17\}$ and temporal width three. Coordinate-specific FiLM scale and shift parameters are bounded by 0.1 and initialized at zero.

At each patch scale, the field operator, future projector, decoder, and batch-normalization parameters are shared across coordinates. During field evaluation, the coordinate axis is folded into the minibatch axis before two-dimensional batch normalization, so normalization statistics are estimated jointly over batch–coordinate examples while the coordinate activations themselves remain separate throughout the nonlinear path.

The coordinate router receives

$$\log(1 + \text{energy}_q), \qquad \log(1 + \text{variance}_q), \qquad \frac{\log(1+H)}{\log(721)},$$

and produces one simplex over coordinates per sample. Field-component, linear-bank, linear–nonlinear, and reference gates are likewise sample-conditioned and broadcast over channels and forecast steps. Scale weights are shared across coordinates. The linear bank operates on the reconstructed signal $\bar{X}$.

Its smoothed component is

$$M(\bar{X}) = 0.35\,\mathrm{MA}_7(\bar{X}) + 0.65\,\mathrm{MA}_{25}(\bar{X}), \tag{26}$$

using replicated boundaries. The analytic trend expert performs ordinary least squares on a centered temporal grid and extrapolates the fitted line to the forecast horizon. The direct field projection is a separate learned linear map, and κ in Equation 10 is a learned scalar passed through a sigmoid.

Training proceeds in stages. The deterministic reference is trained for ten epochs using forecasting MSE together with a 0.03-weighted prefix reconstruction objective. Its scale head is then fitted for two epochs while the forecast mean is frozen. The continuation extension is trained for ten epochs using Equation 12. Both field blocks, future projectors, decoders, coordinate modulation parameters, and continuation gates are updated together with the structured-coordinate modules, while inherited patch encoders, linear-bank parameters, and the reference predictor remain frozen.

Validation MSE selects either the retained reference or one of the extension checkpoints according to Equation 13. The selected checkpoint is then used unchanged for test evaluation.

A.5 Auxiliary Density

For each forecast entry, the auxiliary likelihood uses a two-component Student-t mixture,

$$p_{hc}(y \mid X) = \sum_{j=0}^{1} \omega_j(X)\, t_{\nu_j}(y; \mu_{j,hc}, s_{j,hc}), \qquad \mathcal{L}_{\mathrm{NLL}} = -\frac{1}{HC} \sum_{h,c} \log p_{hc}(Y_{hc} \mid X), \tag{27}$$

with positive scales and $\nu_j > 2$.

The corresponding mixture mean and variance are

$$\widehat{Y} = \sum_j \omega_j \mu_j, \qquad \mathrm{Var}(Y \mid X) = \sum_j \omega_j \frac{\nu_j}{\nu_j - 2} s_j^2 + \sum_j \omega_j (\mu_j - \widehat{Y})^2. \tag{28}$$

The likelihood is used as an auxiliary training signal in Equation 12; all headline forecasting results are evaluated using the selected point prediction $\widehat{Y}$.

B Theoretical Details

Proof of Lemma 1. Let $\alpha = C^{-1}z$, so that $u = S\alpha$. Since $S = \sum_q A_q$ with $A_q \succeq 0$,

$$\ker S = \bigcap_q \ker A_q. \tag{29}$$

Indeed, $x^\top S x = 0$ implies $x^\top A_q x = 0$ for every q, and hence $A_q x = 0$. Since $S^\dagger S\alpha$ is the orthogonal projection of α onto $\mathrm{range}(S)$,

$$\alpha - S^\dagger S\alpha \in \ker S \subseteq \ker A_q.$$

Therefore

$$A_q S^\dagger u = A_q S^\dagger S\alpha = A_q \alpha = m_q.$$

The result holds without requiring S to be full rank. Positive scalar coordinate weights can be absorbed into A_q, yielding the analogous result for a known weighted sum. □

Theorem 1 (Bayes risk under deterministic reduction)**.** *Let Z and Y be square-integrable, let W denote side information, and let $U = A_W(Z)$ be a measurable deterministic reduction. Under squared loss,*

$$\mathcal{R}^*_{U,W} - \mathcal{R}^*_{Z,W} = \mathbb{E}\Big[\|\mathbb{E}[Y \mid Z, W] - \mathbb{E}[Y \mid U, W]\|^2\Big] \geq 0. \tag{30}$$

Equality holds iff the two conditional means agree almost surely.

Proof. The Bayes predictors are the corresponding conditional means. Because $\sigma(U, W) \subseteq \sigma(Z, W)$,

$$\mathbb{E}[Y \mid U, W] = \mathbb{E}[\mathbb{E}[Y \mid Z, W] \mid U, W].$$

Equation 30 then follows from the Pythagorean identity for nested L_2 projections. □

Theorem 1 characterizes deterministic reduction in an unrestricted prediction setting, while Lemma 1 shows that the additive coordinates considered here can remain algebraically recoverable when their fitted covariance operators are retained. The early-fusion experiment therefore isolates the computation implemented by the matched nonlinear forecasting architectures.

Proof of Proposition 1. For each q, Taylor expansion around $\bar{z}$ gives

$$F(z_q) = F(\bar{z}) + J_F(\bar{z})(z_q - \bar{z}) + r_q, \qquad \|r_q\| \leq \frac{M}{2}\|z_q - \bar{z}\|^2. \tag{31}$$

Since $\sum_q p_q(z_q - \bar{z}) = 0$, the weighted first-order terms cancel. Hence

$$\left\| \sum_q p_q F(z_q) - F(\bar{z}) \right\| \leq \sum_q p_q \|r_q\| \leq \frac{M}{2} \sum_q p_q \|z_q - \bar{z}\|^2,$$

which proves Equation 15. For affine F the remainder vanishes. For scalar $F(z) = z^2$,

$$\sum_q p_q F(z_q) - F(\bar{z}) = \sum_q p_q (z_q - \bar{z})^2,$$

the weighted variance. □

Proposition 2 (Validity and residual accounting)**.** *The specified covariance families yield positive-semidefinite Q_q and positive-definite C_g. Equation 21 produces ordered interior inducing timestamps, and*

$$\sum_q m_{gq} = z - (\sigma_n^2 + \epsilon) C_g^{-1} z. \tag{32}$$

Proof. Periodic, Matérn, and exponential kernels are positive semidefinite, and products preserve positive semidefiniteness. For any v,

$$v^\top Q_q v = (K_{U_q T} v)^\top (K_{U_q U_q} + \epsilon I)^{-1} (K_{U_q T} v) \geq 0.$$

Thus a nonnegative weighted sum of Q_q plus $(\sigma_n^2 + \epsilon) I$ is positive definite. Positive inducing gaps yield strictly increasing cumulative fractions in $(0, 1)$. The residual identity follows directly from Equation 2. □

Convex aggregation. For simplex weights w_j, Jensen's inequality gives

$$\left\| Y - \sum_j w_j Y_j \right\|_F^2 \leq \sum_j w_j \|Y - Y_j\|_F^2. \tag{33}$$

The difference between the right- and left-hand sides is

$$\frac{1}{2} \sum_{j,k} w_j w_k \|Y_j - Y_k\|_F^2. \tag{34}$$

The identity applies pointwise to prefix-conditioned gates. Equation 13 additionally retains the reference predictor as an explicit validation candidate.

Coordinate symmetries. A simultaneous relabeling of covariance families, group allocations, coordinate modulation parameters, and coordinate-gate outputs leaves the forecast unchanged. Related parameterizations can therefore represent the same forecasting computation. This is consistent with our interpretation of the latent variables as structured temporal coordinates rather than uniquely identified physical processes.

Computational scaling. For fixed L, H, Q, G, M, D_z, D and local kernel sizes, the dominant continuation cost scales linearly with the number of channels C. A field layer requires

$$O\left(BQC \sum_{s \in \mathcal{S}} P_s \left[D \sum_k 3k + D^2 \right] \right), \tag{35}$$

in addition to future projection and decoding. The structured-coordinate frontend uses $O(QM^3)$ inducing factorizations and $O(GL^3)$ observed-grid factorizations, followed by batched solves scaling with BCD_z. Because no dense $C \times C$ covariance matrix is formed, channel growth enters these operations through the batched applications rather than through channel-wise matrix factorization.

C Data and Protocol

ETT follows the established chronological splits introduced with Informer (Zhou et al., 2021); the remaining datasets use chronological 70/10/20% splits. All tasks are multivariate-to-multivariate with input length $L = 96$ and horizons $H \in \{96, 192, 336, 720\}$. No calendar covariates are supplied.

For the controlled fixed-budget evaluation, each task uses 2,048 sampled training windows and 1,024 validation windows. Window indices are fixed across compared methods and initialization seeds,

Table 6: Benchmark characteristics before window extraction.

Dataset	Observations	Channels	Cadence	Split
ETTh1, ETTh2	17,420 each	7	hourly	12/4/4 months
ETTm1, ETTm2	69,680 each	7	15 minutes	12/4/4 months
Weather	52,696	21	10 minutes	70/10/20%
Electricity	26,304	321	hourly	70/10/20%
Traffic	17,544	862	hourly	70/10/20%

Table 7: Information used at each stage of training and evaluation.

Operation	Information used
Standardization	Complete training split; statistics frozen thereafter.
Encoding and routing	Observed forecast prefix only.
Gradient optimization	Sampled training windows and their forecast targets.
Configuration and checkpoints	Sampled validation set and validation MSE.
Final evaluation	All valid test windows after model selection.

while every valid test window is evaluated. Forecast errors are averaged over windows, forecast steps, and channels.

Dataset standardization is fitted once using the complete training split and then frozen for validation and test. Prefix normalization performed inside LATENTFLOW is reversed before evaluation. Dataset-level LATENTFLOW configurations are selected using validation MSE and then held fixed across seeds; final settings are reported in Table 5.

D COMPARED IMPLEMENTATIONS

All baselines are rerun under the common fixed-window protocol described in Appendix C. We use the recommended configurations from the corresponding papers, official implementations, and TSLib/TSLibrary where available, including dataset- and horizon-specific settings. DLinear uses moving-average length 25. iTransformer and TimeMixer use width 128, feed-forward width 256, two layers, and dropout 0.1; iTransformer uses eight attention heads. TimeMixer uses one factor-two average-downsampling level and channel-independent decomposition. VPNet uses patch length 16, width 256, two local 3×3 blocks, dropout 0.05, and reconstruction weight 0.1. xCPD is implemented as its graph-spectral routing module coupled to a DLinear host. TimePro follows its released hyper-state computation and dataset–horizon configurations.

All models use validation-MSE checkpointing with patience three. Wide-dataset micro-batches are capped at four, with batch size two for xCPD on Traffic. For xCPD, the DLinear host is fitted for ten epochs, followed by plug-in training with maxima of 10 epochs on ETT/Weather, 15 on Electricity, and 30 on Traffic. Its hidden widths are 256 for ETTh1, ETTh2, ETTm1, and Weather; 128 for ETTm2; 512 for Electricity; and 2,048 for Traffic.

For LATENTFLOW, AdamW uses learning rate 7×10^{-4} and weight decay 10^{-4}, with effective batch size 16 and gradient clipping at 5. Training consists of a 10-epoch reference stage, a 2-epoch scale-head stage, and a 10-epoch continuation stage. Final dataset-level LATENTFLOW configurations are listed in Table 5.

E CROSS-SEED CONFIRMATION

Table 10 uses seeds 42–46. Overall MSE is 0.34796 for LATENTFLOW versus 0.35136 for TimePro; MAE is 0.35079 versus 0.35243. LATENTFLOW has lower dataset-average MSE on five of seven datasets; TimePro leads ETTh1 and ETTm1. For MAE, TimePro leads ETTm1 and Electricity.

Across 140 paired cells, LATENTFLOW wins 94 MSE and 88 MAE comparisons. Its relative MSE gain is 1.38%, with source–seed interval $[0.74, 2.14]\%$ and source-family interval $[0.14, 3.37]\%$. The

Table 8: Architectural distinctions among the evaluated forecasting methods.

Method	Forecast state	Main interaction	Representation–operator interface
DLinear (2023)	Trend and residual signals	Per-channel linear maps	Decomposes first and forecasts components linearly.
TimeMixer (2024)	Decomposed multiscale states	Across-scale mixing	Mixes decomposed states across resolutions.
iTransformer (2024)	Variable tokens	Attention across variables	Embeds temporal history within each variable token.
VPNet (2026)	Variate–patch field	Local variable neighborhoods	Applies nonlinear continuation to a single field state.
xCPD (2026)	Host and dependency states	Graph-spectral routing	Routes dependency information around a host predictor.
TimePro (2025)	Time- and variable-aware hyper-state	Hyper-state propagation	Integrates temporal and variable context within a shared hyper-state.
LatentFlow (Ours)	Explicit structured-coordinate fields	Shared local variate–patch operators	Retains coordinates through horizon decoding and fuses futures late.

Table 9: Optimization settings used for the baseline reruns.

Method	Learning rate	Weight decay	Epochs	Schedule
DLinear (2023)	10^{-4}	0	10	none
iTransformer (2024)	10^{-4}	0	10	none
TimeMixer (2024)	10^{-4}	0	10	none
VPNet (2026)	10^{-3}	10^{-4}	10	none
TimePro (2025)	5×10^{-5}–5×10^{-4}	10^{-4}	10	cosine
xCPD (2026)	10^{-4}	10^{-4}	10–30	none

five source families combine ETTh1 with ETTm1 and ETTh2 with ETTm2. Leaving out any one family retains a positive MSE gain; the smallest is 0.72% when Weather is omitted. MAE gain is 0.56%, with source–seed interval $[0.16, 0.97]\%$ and source-family interval $[-0.28, 1.42]\%$.

E.1 Statistical Definitions

Relative improvement for cell i is $100(1 - e_i^{\mathrm{LF}}/e_i^{\mathrm{comparator}})$. Arithmetic MSE/MAE means weight dataset–horizon tasks equally, not by test-window or channel count. Rank is computed within task before averaging. Table formatting uses unrounded values; equal displayed decimals can have different ranks.

Bootstrap intervals use 20,000 resamples. The primary source-family bootstrap resamples five groups while retaining every associated cadence, horizon, and seed: ETT station 1, ETT station 2, Weather, Electricity, and Traffic. The sensitivity analysis resamples 25 source–seed clusters with their horizons retained. These intervals describe variation over the observed sources, not an unlimited population of independent datasets. Table 12 includes every seed-42 comparison and the MAE interval against TimePro that crosses zero. Two-sided Wilcoxon tests use 28 task differences with Holm adjustment within metric and are descriptive because horizons share data.

Table 10: Five-seed confirmation (42–46). Average horizons within each seed first, then report mean and sample SD over seeds. Overall first averages all 28 tasks. SD is not across horizons.

Dataset	Metric	LatentFlow (Ours)	TimePro (2025)
ETTh1	MSE	0.4611±0.0022	**0.4606±0.0008**
	MAE	**0.4477±0.0014**	0.4495±0.0005
ETTh2	MSE	**0.3814±0.0051**	0.3824±0.0011
	MAE	**0.4047±0.0030**	0.4068±0.0010
ETTm1	MSE	0.4129±0.0046	**0.4109±0.0014**
	MAE	0.4171±0.0019	**0.4123±0.0009**
ETTm2	MSE	**0.2846±0.0011**	0.2925±0.0003
	MAE	**0.3280±0.0004**	0.3346±0.0003
Weather	MSE	**0.2516±0.0006**	0.2645±0.0003
	MAE	**0.2794±0.0007**	0.2854±0.0002
Electricity	MSE	**0.1843±0.0014**	0.1867±0.0002
	MAE	0.2760±0.0010	**0.2756±0.0001**
Traffic	MSE	**0.4599±0.0025**	0.4619±0.0004
	MAE	**0.3026±0.0014**	0.3029±0.0002
Overall	MSE	**0.3480±0.0016**	0.3514±0.0002
	MAE	**0.3508±0.0009**	0.3524±0.0001

Table 11: Additional five-seed comparisons (horizon-averaged mean ± sample SD). Each comparator was chosen post-hoc as the strongest non-TimePro method in initial seed-42 results.

Dataset	Comparator	LF MSE	Comp. MSE	LF MAE	Comp. MAE
ETTh1	iTransformer (2024)	0.4611±0.0022	**0.4562±0.0018**	**0.4477±0.0014**	0.4499±0.0010
ETTh2	TimeMixer (2024)	**0.3814±0.0051**	0.3819±0.0004	**0.4047±0.0030**	0.4050±0.0002
ETTm1	xCPD (2026)	0.4129±0.0046	**0.4108±0.0015**	0.4171±0.0019	**0.4127±0.0014**
ETTm2	TimeMixer (2024)	**0.2846±0.0011**	0.2934±0.0003	**0.3280±0.0004**	0.3365±0.0005
Weather	VPNet (2026)	**0.2516±0.0006**	0.2555±0.0011	**0.2794±0.0007**	0.2821±0.0011
Electricity	VPNet (2026)	**0.1843±0.0014**	0.1997±0.0009	**0.2760±0.0010**	0.2901±0.0010
Traffic	VPNet (2026)	**0.4599±0.0025**	0.4933±0.0018	**0.3026±0.0014**	0.3309±0.0020

Table 12: Seed-42 paired comparisons. Positive gain favors LatentFlow. Intervals resample five source families, retaining their datasets and all four horizons. Two-sided Wilcoxon tests use 28 task pairs with Holm correction over six comparisons within each metric; these tests are descriptive.

Comparator	Metric	W/T/L	Gain (%)	95% CI	p_{Holm}
TimePro (2025)	MSE	23/0/5	1.87	[0.35, 3.72]	0.0029
VPNet (2026)	MSE	28/0/0	6.67	[4.72, 7.86]	4.47e-08
xCPD (2026)	MSE	24/0/4	12.02	[3.15, 20.72]	2.39e-05
iTransformer (2024)	MSE	24/0/4	6.97	[2.60, 14.03]	2.59e-05
TimeMixer (2024)	MSE	25/0/3	9.20	[3.23, 19.94]	3.28e-06
DLinear (2023)	MSE	26/0/2	14.99	[5.66, 24.33]	3.28e-06
TimePro (2025)	MAE	18/0/10	0.80	[-0.12, 1.75]	0.0247
VPNet (2026)	MAE	28/0/0	5.11	[3.28, 6.76]	4.47e-08
xCPD (2026)	MAE	23/0/5	10.21	[3.48, 17.09]	5.04e-06
iTransformer (2024)	MAE	26/0/2	5.20	[1.65, 11.16]	9.99e-06
TimeMixer (2024)	MAE	25/0/3	6.51	[1.75, 15.09]	2.21e-05
DLinear (2023)	MAE	27/0/1	13.38	[5.71, 20.67]	5.22e-07

F Final-Fit and Representation Audits

Checkpoint replay and validation selection. Each archived final checkpoint is reloaded and evaluated on the same test windows using the frozen reference, learned extension, and validation-selected predictor defined in Equation 13. All 140 five-seed fits are replayed, and every fit with an exported source metric matches its recorded value to numerical precision. Validation selects the learned extension in 108 of 140 fits, confirming that the reported predictor typically uses the learned continuation rather than defaulting to the frozen reference.

Coordinate-prior controls. We perturb sensor-group priors, process-family assignments, and initial temporal scales while preserving the same training and validation-selection protocol. The resulting aggregate changes are small, and all source-family intervals include zero (Table 13). These controls support the interpretation that the forecasting result is not driven by privileged coordinate labels or a particular prior initialization.

Deterministic representation control. To separate representation family from continuation, we replace the GP-derived coordinates with a deterministic multibranch construction using learned temporal filters. The control preserves the number of branches, reconstructs the same structural state, and retains the complete continuation and validation-selection architecture. Its aggregate point-forecasting accuracy is nearly identical to the GP-derived version: MSE differs by 0.17%, and the 28 task means split evenly, 14–14 (Table 14). This supports the broader claim that the main effect arises from preserving an explicit coordinate axis through nonlinear continuation rather than from the specific stochastic parameterization.

Control separation. We also verify directly that the early-fusion and causal-state-only controls implement distinct predictors. Across all 140 fits, the two controls are neither bitwise identical nor numerically all-close; their mean absolute prediction difference is 0.0561. Table 15 reports the complete comparison. All intervention checkpoints are evaluated using their stored validation selection, matching the prediction rule used in the main experiments.

G Patch-Resolution Control and Factorial Analysis

Patch-resolution control. The released model evaluates multiple patch resolutions within each retained coordinate field to provide complementary temporal receptive fields. Table 16 compares this configuration with a matched single-resolution control over all 28 tasks and five seeds. The aggregate difference is small, while the larger gains concentrate on Electricity and Traffic. The single-resolution baseline selects its patch scale by validation MSE for each task and seed.

Across horizons 96, 192, 336, and 720, the relative MSE changes of the multi-resolution model against the single-resolution control are $+0.03\%$, $+0.27\%$, -0.40%, and -0.32%, respectively. These results indicate that multi-resolution processing acts primarily as a supporting architectural choice rather than the source of the principal forecasting effect.

Factorial analysis. Tables 17 and 18 report the complete three-seed $2 \times 2 \times 2$ factorial experiment over coordinate preservation, multi-resolution processing, and the linear bank. After marginalizing over the other factors, preserving the coordinate axis improves both MSE and MAE, with clustered intervals below zero. The linear bank provides a smaller complementary improvement, while patch resolution shows no aggregate main effect. The factorial experiment therefore independently identifies componentwise continuation as the dominant architectural factor.

H Gate and Separator Controls

Gate diagnostics. Table 19 examines the relationship between sample-conditioned gate weights and the standalone accuracy of their corresponding branches. The learned gates reflect joint mixture behavior rather than a direct ranking of individual branch quality; accordingly, their role is evaluated primarily through controlled architectural removals and the factorial experiments in Appendix G.

Table 13: Prior-dependence controls over 84 dataset–horizon–seed cells (seeds 42–44). Positive change favors the final configuration. Confidence intervals resample five source families. B/T/W denotes control better / tie / control worse.

Control	MSE	Δ MSE (%)	95% CI	B/T/W
No sensor-group priors	0.34759	+0.046	[-0.09, 0.24]	24/21/39
Permuted process families	0.34732	-0.134	[-0.54, 0.09]	30/20/34
Neutral process scales	0.34705	-0.221	[-0.68, 0.13]	39/18/27

Table 14: Five-seed matched representation control. The deterministic multibranch model replaces the sparse GP-derived coordinates with bounded learned temporal filters while preserving branch count, exact latent reconstruction, continuation, training, and validation selection. Each entry averages four horizons and five seeds. Lower is better; the final column gives LatentFlow/control wins over the four horizon means.

Dataset	LatentFlow MSE	Deterministic MSE	LatentFlow MAE	Deterministic MAE	MSE W/L
ETTh1	0.4611	**0.4610**	**0.4477**	0.4481	1/3
ETTh2	**0.3814**	0.3828	**0.4047**	0.4055	3/1
ETTm1	**0.4129**	0.4130	**0.4171**	0.4177	3/1
ETTm2	0.2846	**0.2843**	0.3280	**0.3279**	2/2
Weather	**0.2516**	0.2528	**0.2794**	0.2804	4/0
Electricity	0.1843	**0.1828**	0.2760	**0.2740**	1/3
Traffic	0.4599	**0.4552**	0.3026	**0.2989**	0/4
Overall	0.3480	**0.3474**	0.3508	**0.3504**	14/14

Separator control. We additionally augment VPNet with the same structured separator while retaining VPNet's standard single-state nonlinear continuation. Table 20 reports the resulting comparison. Adding the separator without preserving its coordinate axis through the forecasting operator does not reproduce the gains of LATENTFLOW. This provides an additional control for the representation–continuation interpretation: the benefit arises from how structured coordinates are consumed by the nonlinear predictor, rather than from introducing the structured frontend alone.

I CHANNEL-ORDER SENSITIVITY

We evaluate sensitivity to the supplied variable ordering by retraining the model under five channel permutations, applied consistently to training, validation, and test data. At horizon 720, the resulting mean MSE change is $+9.84\%$ on Electricity and $+1.96\%$ on Traffic, with ranges 9.15–10.39% and 1.82–2.05%, respectively.

The effect is consistent with the local variate operators used by the continuation field: changing channel order changes the neighborhoods seen by those operators. Thus the supplied ordering forms part of the model input structure. Incorporating learned or order-robust variable geometry is a natural extension of the current architecture.

J EFFICIENCY AND CHANNEL SCALING

Table 22 profiles all seven methods over the 28 forecasting tasks at batch size one. FLOPs are profiler estimates, while latency is measured separately on an NVIDIA GH200 120GB. Reported parameter counts include the components required by the evaluated prediction path.

We additionally measure empirical channel scaling on Traffic at horizon 720, using subsets from 32 to 862 channels while preserving their original relative ordering. At 862 channels, LATENTFLOW reaches 99.85 ms batch-four latency compared with 504.72 ms for TimePro, while its parameter

Table 15: Prediction-level comparison of early fusion and causal-state-only controls over five seeds and four horizons. No pair is bitwise identical or numerically all-close ($r = 10^{-6}$, $a = 10^{-7}$).

Dataset	Fits	Identical/all-close	Mean $\lvert\Delta\widehat{Y}\rvert$
ETTh1	20	0/0	0.0645
ETTh2	20	0/0	0.0545
ETTm1	20	0/0	0.0741
ETTm2	20	0/0	0.0607
Weather	20	0/0	0.0229
Electricity	20	0/0	0.0566
Traffic	20	0/0	0.0594
Overall	140	0/0	0.0561

Table 16: Five-seed patch-resolution analysis. The selected-configuration column reports the released multi-resolution model; the single-resolution control retains process preservation and the linear bank but uses one validation-selected resolution per task. Δ is the single-resolution MSE change relative to the released model, so positive values favor the released configuration.

Dataset	Selected-config. MSE	Single-resolution MSE	Δ MSE (%)
ETTh1	0.4611	**0.4576**	−0.765
ETTh2	0.3814	**0.3802**	−0.300
ETTm1	0.4129	**0.4124**	−0.116
ETTm2	0.2846	**0.2829**	−0.594
Weather	0.2516	**0.2502**	−0.561
Electricity	**0.1843**	0.1856	+0.684
Traffic	**0.4599**	0.4623	+0.520
Overall	0.3480	**0.3473**	−0.106

count remains approximately 2.02M. Peak inference memory is 939.4 MiB for LATENTFLOW and 211.4 MiB for TimePro.

Each channel-count configuration is trained independently. Runtime over a finite hardware range also reflects accelerator utilization, so the empirical latency curve need not be exactly proportional to channel count. The corresponding architectural scaling statement is that, for fixed temporal dimensions and model widths, continuation cost grows linearly with the number of channels, as derived in Appendix B.

Table 17: Complete three-seed factorial cells over 28 tasks. Selected and single denote patch-resolution settings; early and preserved specify where the process axis is reduced.

Resolution	Linear bank	Process axis	MSE	MAE
Selected	Off	Early	0.3636	0.3682
Selected	Off	Preserved	0.3508	0.3531
Selected	On	Early	0.3583	0.3633
Selected	On	Preserved	0.3476	0.3505
Single	Off	Early	0.3608	0.3662
Single	Off	Preserved	0.3486	0.3514
Single	On	Early	0.3629	0.3661
Single	On	Preserved	**0.3469**	**0.3493**

Table 18: Balanced three-seed factorial contrasts. Factors are coded as $-1/+1$: single/multiscale, bank off/on, and early/preserved continuation. Thus a negative main effect favors the $+1$ level; interaction signs compare the positive and negative products of the corresponding factor codes. Intervals resample datasets with their horizons and seeds and are unadjusted descriptive intervals.

Contrast	MSE diff.	95% CI	MAE diff.	95% CI
scale	+0.000272	[-0.003187, +0.003859]	+0.000533	[-0.002022, +0.003146]
bank	-0.002018	[-0.004503, +0.000469]	-0.002436	[-0.004146, -0.000560]
continuation	-0.012937	[-0.026758, -0.000432]	-0.014867	[-0.030608, -0.000667]
scale × bank	-0.002238	[-0.004121, -0.000223]	-0.001349	[-0.002892, +0.000134]
scale × continuation	+0.001193	[-0.000053, +0.002627]	+0.000947	[+0.000098, +0.001917]
bank × continuation	-0.000427	[-0.001091, +0.000193]	+0.000116	[-0.000619, +0.000891]
scale × bank × continuation	+0.001522	[-0.000064, +0.003215]	+0.001088	[-0.000187, +0.002668]

Table 19: Gate diagnostics on seven dataset–horizon pairs, seed 42. Weight–quality correlation measures agreement with realized expert quality; it does not establish causal or physical meaning.

Dataset	H	Bank weight–quality r	Field weight–quality r
ETTh1	720	0.301	-0.645
ETTh1	96	0.276	0.221
ETTm1	336	0.227	-0.560
ETTm1	720	0.340	-0.338
Electricity	336	0.625	-0.730
Traffic	192	0.561	0.143
Weather	720	0.297	-0.349

Table 20: Stochastic frontend control, seed 42, MSE averaged over four horizons. The VPNet separator control uses early fusion and no exchange. The LatentFlow column is the final no-exchange model; it is a contextual comparison, not a parameter-matched VPNet removal.

Dataset	VPNet (2026)	VPNet + separator	LatentFlow (Ours)
ETTh1	0.4877	0.5229	**0.4580**
ETTh2	0.4042	0.3954	**0.3732**
ETTm1	0.4487	0.4455	**0.4168**
ETTm2	0.3055	0.3027	**0.2841**
Weather	0.2561	0.2619	**0.2511**
Electricity	0.1996	0.2366	**0.1823**
Traffic	0.4956	0.5171	**0.4585**
Overall	0.3711	0.3832	**0.3463**

Table 21: Channel-order sensitivity: horizon 720, training seed 42, five random orders per dataset. Models are retrained and evaluated in the same permuted order. Ranges are across permutations, not training seeds.

Dataset	Original MSE	Permuted MSE	Degradation (%)	Range (%)
Electricity	0.2171	0.2385	9.84	[9.15, 10.39]
Traffic	0.4915	0.5011	1.96	[1.82, 2.05]

Table 22: Profiled cost averaged over 28 tasks on an NVIDIA GH200 120GB, batch size one. Parameter counts include frozen parameters. FLOP estimates omit unsupported operators; latency is measured independently. This sweep uses the no-exchange model.

Model	Params (M)	GFLOPs	Infer. ms
LatentFlow (Ours)	1.64	11.00	25.21
TimePro (2025)	7.24	5.93	58.24
VPNet (2026)	0.67	1.79	1.21
xCPD (2026)	4.48	11.70	2.87
iTransformer (2024)	0.32	0.19	1.99
TimeMixer (2024)	0.23	10.74	3.36
DLinear (2023)	0.07	0.02	0.34

Table 23: Real Traffic channel scaling at horizon 720. Channels are selected by training-split variance. Latency and memory use batch size four; FLOPs are per sample. These are independently retrained scaling fits and are used only for computational measurements.

Channels	Model	Params (M)	GFLOPs	ms/batch	Peak MiB
32	LatentFlow (Ours)	2.017	2.98	31.17	108.4
32	TimePro (2025)	14.738	1.17	20.47	178.3
64	LatentFlow (Ours)	2.017	5.94	32.43	144.6
64	TimePro (2025)	14.738	2.35	38.43	179.2
128	LatentFlow (Ours)	2.018	11.85	34.63	205.3
128	TimePro (2025)	14.738	4.70	74.40	182.5
256	LatentFlow (Ours)	2.018	23.67	37.17	335.4
256	TimePro (2025)	14.738	9.40	140.97	186.4
512	LatentFlow (Ours)	2.019	47.32	52.43	587.7
512	TimePro (2025)	14.738	18.80	278.42	198.9
862	LatentFlow (Ours)	2.020	79.64	99.85	939.4
862	TimePro (2025)	14.738	31.65	504.72	211.4