COMPASS: DEMAND-GUIDED LATENT COMPRESSION FOR HIGH-DIMENSIONAL TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

High-dimensional time series forecasting requires capturing cross-variable dependencies while maintaining computational efficiency. Channel-independent methods scale well but overlook cross-variable dependencies, whereas channel-dependent methods become increasingly expensive as the number of variables grows. Channel compression offers a promising solution, yet existing approaches apply a shared compression scheme without accounting for heterogeneous representational demands across variables. To address this limitation, we propose ComPass, a demand-guided latent compression framework that allocates limited latent capacity according to variable demands. Specifically, a demand analyzer evaluates each variable through a lightweight shared temporal bottleneck, and a budget router converts the resulting scores into continuous budget priors. Attention biases derived from demand guide learnable latent queries, while an alignment objective encourages attention usage to follow the budget priors. To preserve information useful for forecasting under differentiated capacity allocation, we further introduce a budget-conditioned prediction preservation mechanism. During training, this mechanism employs an auxiliary head to generate forecasts from recovered variable representations and their corresponding budget embeddings. The auxiliary forecasts are aligned with outputs from the main forecasting head, which is supervised by the forecasting targets. This prediction consistency favors compressed representations that retain information useful for forecasting under the allocated budgets, without adding inference overhead. Experiments on 16 high-dimensional forecasting benchmarks demonstrate that ComPass achieves strong forecasting performance while handling time series with up to 20,000 variables.

1 INTRODUCTION

Multivariate time series forecasting aims to predict the future evolution of systems characterized by interrelated variables. It has been widely applied in fields such as energy networks, traffic monitoring, and financial markets (Cirstea et al., 2022; Fang et al., 2021; Kieu et al., 2025; Wu et al., 2024; Hu et al., 2026b; Wu et al., 2025b; Campos et al., 2021). As data acquisition technologies continue to advance, modern systems can simultaneously record thousands or even tens of thousands of variables, such as measurements from distributed sensors, connected devices, or financial assets (Granger & Newbold, 2014; Wu et al., 2023b; Yin et al., 2021). Accurate forecasts of these high-dimensional systems provide important support for resource allocation and risk management.

Modeling high-dimensional time series requires handling an expanding space of potential cross-variable interactions. Two fundamental paradigms are channel-independent and channel-dependent modeling. As illustrated in Figure 1(a), channel-independent methods process each variable separately and scale efficiently with the variable dimension (Nie et al., 2022; Dai et al., 2024; Zeng et al., 2023; Hu et al., 2026a), but do not explicitly capture cross-variable dependencies. In contrast, channel-dependent methods directly model such dependencies (Figure 1(b)) (Zhou et al., 2021; Wu et al., 2023a; Liu et al., 2023; Huang et al., 2023; Zhang & Yan, 2023), but incur rapidly increasing computational and memory costs as the number of variables grows. Recent variants alleviate this cost by restricting interactions to channel clusters (Qiu et al., 2025b; Wu et al., 2025a; Hu et al., 2025). However, reliable cluster assignment becomes increasingly challenging as the number of variables increases. This raises a question: **How can variable dependencies be captured efficiently in high-dimensional time series forecasting?**

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.

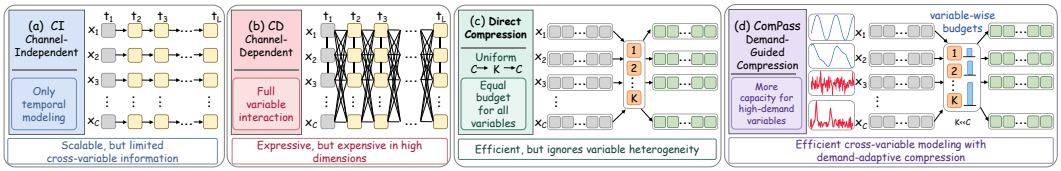


Figure 1: Channel modeling paradigms for forecasting. (a) Channel-independent modeling scales efficiently but lacks explicit cross-variable interaction. (b) Channel-dependent modeling captures such interactions at growing computational and memory costs. (c) Uniform latent compression improves efficiency but overlooks heterogeneous variable demands. (d) ComPass derives variable-wise budget priors from these demands to guide latent compression and cross-variable modeling.

Channel compression offers a solution by mapping variables into compact latent representations before modeling cross-variable interactions, thereby reducing the interaction dimension. However, existing methods (Ni et al., 2025) apply a shared compression scheme without accounting for heterogeneous variable demands (Figure 1(c)). Under a common compact representation constraint, variables may differ in how adequately their temporal information can be represented. Variables that incur greater information loss have higher relative representational demands and may require more latent capacity. We refer to the total modeling capacity for allocation across variables during latent compression as the compression budget. Because this budget is limited, uniform compression may underrepresent high-demand variables while allocating redundant capacity to low-demand ones. Thus, **the first challenge** is how to characterize heterogeneous variable-wise representational demands and translate them into differentiated capacity allocation during latent compression.

Moreover, differentiated allocation determines how much capacity each variable receives, but not which information is preserved. When many variables are compressed into a few latent representations, forecasting-relevant information may still be diluted. In computer vision, task-aware compression methods incorporate downstream classification, detection, or tracking objectives to preserve task-relevant content (Ge et al., 2024; Park et al., 2025). However, existing task-aware compression objectives are typically defined at the sample level and do not directly address variable-wise capacity allocation under heterogeneous representational demands. Consequently, patterns that dominate the overall forecasting objective may be preferentially retained, while complementary predictive cues from other variables may be diluted during compression and recovery. Therefore, **the second challenge** is how to preserve information useful for forecasting throughout compression and recovery under differentiated capacity allocation.

To address these challenges, we propose **ComPass**, a demand-guided latent compression framework for high-dimensional time series forecasting. Its *demand analyzer* evaluates each variable through a lightweight shared temporal bottleneck and uses the reconstruction discrepancy as a comparable demand score. The *adaptive budget router* converts these scores into continuous budget priors. The resulting routing signals guide learnable latent queries through demand-based attention biases, while budget–usage alignment encourages the realized attention usage to follow the intended capacity allocation (Figure 1(d)). To preserve forecasting-relevant information under differentiated capacity allocation, we further introduce a *budget-conditioned prediction preservation mechanism*. Specifically, an auxiliary head generates budget-conditioned forecasts from the recovered variable representations and their corresponding budget embeddings. The budget-conditioned forecasts are aligned with the outputs of the target-supervised main forecasting head. The prediction consistency objective favors compressed representations that retain forecasting-relevant information under the allocated budgets. The contributions are summarized as follows:

- We propose a demand-guided latent compression method that converts heterogeneous variable demands into continuous budget priors. By aligning compression with these budgets, it enables differentiated allocation of limited capacity across variables.
- We propose a budget-conditioned prediction preservation objective that aligns auxiliary forecasts with outputs from the target-supervised main forecasting head. By using forecasting utility to evaluate recovered representations, the objective encourages retention of forecasting-relevant information under differentiated capacity allocation.
- Extensive experiments on 16 high-dimensional forecasting benchmarks demonstrate that ComPass achieves competitive or superior performance while maintaining practical scala-

bility to time series with up to 20,000 variables. Further analyses validate the effectiveness of demand-guided capacity allocation and budget-conditioned prediction preservation.

2 Related Work

2.1 Time Series Forecasting

Existing deep learning-based forecasting methods adopt different paradigms to capture temporal patterns. Recurrent and convolutional models characterize sequential evolution and local temporal dependencies (Lai et al., 2018; Bai et al., 2018). Transformer-based methods model long-range dependencies through attention, decomposition, or autocorrelation mechanisms (Zhou et al., 2021; Wu et al., 2021). Patch-based models, such as PatchTST (Nie et al., 2022) and LiPFormer (Wang et al., 2025), divide a sequence into temporal patches to capture local patterns while reducing the effective sequence length. Other approaches exploit periodicity and temporal structures through two-dimensional variation modeling (Wu et al., 2023a), multiscale mixing (Wang et al., 2024), or frequency-domain representations (Zhou et al., 2022; Ye et al., 2025). Meanwhile, linear and MLP-based models provide efficient alternatives for temporal forecasting (Zeng et al., 2023; Chen et al., 2023). As real-world monitoring systems increasingly produce thousands or even more variables, forecasting models need to address not only temporal dependency learning but also scalable and effective modeling of the variable dimension. This motivates recent studies on channel modeling in multivariate time series forecasting.

2.2 Channel Modeling for High-Dimensional Forecasting

Channel-independent methods model each variable separately and scale efficiently with the variable dimension, but do not explicitly capture cross-variable dependencies (Nie et al., 2022; Zeng et al., 2023). Channel-dependent methods instead model variable interactions through attention, channel mixing, or graph structures (Zhou et al., 2021; Wu et al., 2023a; Liu et al., 2023; Wu et al., 2020; Huang et al., 2023). However, their computational and memory costs can increase rapidly with the number of variables. Recent approaches reduce these costs through channel grouping, clustering, or selective interaction (Qiu et al., 2025b; Wu et al., 2025a). Although these strategies avoid modeling all pairwise relationships, identifying effective interaction structures becomes increasingly challenging as the variable space expands. Channel compression offers another solution by mapping numerous variable representations into a compact latent space before interaction modeling (Ni et al., 2025). Existing methods generally employ shared compression schemes without explicitly distinguishing heterogeneous representational demands across variables. Such uniform compression may underrepresent high-demand variables while allocating redundant capacity to low-demand ones. In computer vision, task-aware compression methods incorporate downstream objectives to preserve task-relevant content (Ge et al., 2024; Park et al., 2025). When transferred to high-dimensional time series, however, these methods assess information utility for the multivariate input as a whole, without considering that individual variables are represented and recovered under different budgets. As a result, patterns that dominate the overall forecasting objective may be preferentially retained, while complementary predictive cues from other variables may be diluted during compression and recovery. In contrast, ComPass couples demand-guided capacity allocation with budget-conditioned prediction preservation, enabling variable demands to guide both latent capacity allocation and forecasting-relevant information retention.

3 Methodology

Let $\mathbf{X} = [\mathbf{x}_{t-L+1}, \ldots, \mathbf{x}_t]^\top \in \mathbb{R}^{L\times C}$ denote a historical multivariate time series, where L is the lookback length, C is the number of variables, and $\mathbf{x}_t \in \mathbb{R}^C$ contains the observations of all variables at time t. Given $\mathbf{X}$, the forecasting task is to predict the subsequent S time steps $\mathbf{Y} = [\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+S}]^\top \in \mathbb{R}^{S\times C}$. We learn a forecasting model f_θ that produces:

$$\widehat{\mathbf{Y}} = f_\theta(\mathbf{X}), \qquad f_\theta : \mathbb{R}^{L\times C} \to \mathbb{R}^{S\times C}, \tag{1}$$

where $\widehat{\mathbf{Y}}$ denotes the predicted future sequence. We focus on high-dimensional settings in which C may reach thousands or more.

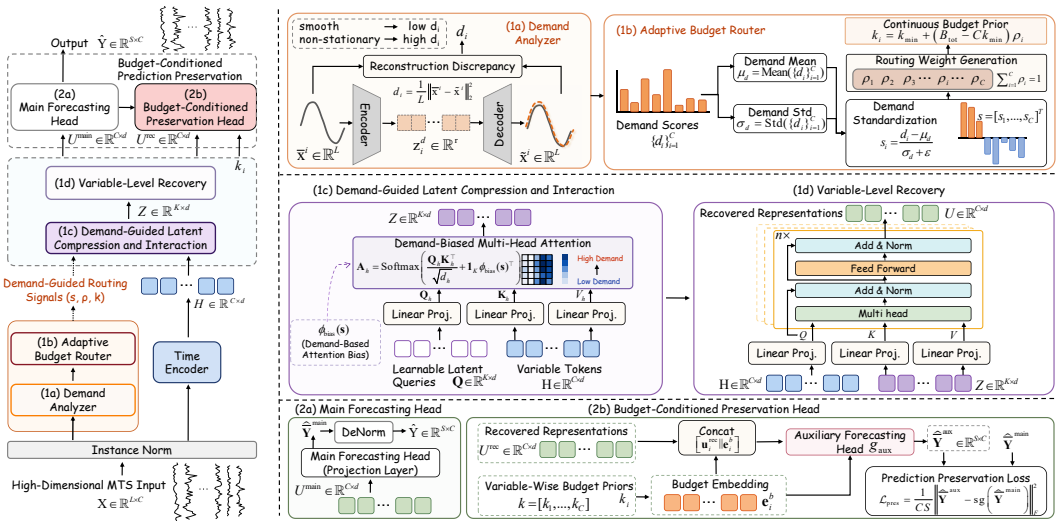


Figure 2: **Overall framework of ComPass.** A demand analyzer estimates variable-wise demands through a shared bottleneck, and an adaptive budget router converts them into continuous budget priors. The resulting signals guide latent compression and interaction, followed by variable-level recovery and residual fusion for forecasting. During training, a budget-conditioned preservation branch aligns auxiliary forecasts with stop-gradient main forecasts to preserve forecasting-relevant information along the latent pathway.

3.1 Overall Framework

To efficiently capture cross-variable dependencies in high-dimensional time series, we propose **ComPass**, as illustrated in Figure 2. ComPass models variable interactions through a compact latent space and adapts the compression process to heterogeneous variable demands.

ComPass consists of two coupled components: demand-guided latent compression and budget-conditioned prediction preservation. Given an input $\mathbf{X} \in \mathbb{R}^{L\times C}$, we first apply instance normalization independently to each variable, producing the normalized input $\overline{\mathbf{X}}$. A shared temporal encoder then maps $\overline{\mathbf{X}}$ into variable-wise temporal representations $\mathbf{H} \in \mathbb{R}^{C\times d}$. Meanwhile, the demand analyzer passes each normalized variable through a shared bottleneck and obtains the variable-wise demand scores $\{d_i\}_{i=1}^{C}$. The adaptive budget router further converts these scores into standardized demand signals $\{s_i\}_{i=1}^{C}$, routing weights $\{\rho_i\}_{i=1}^{C}$, and continuous budget priors $\{k_i\}_{i=1}^{C}$.

Guided by these routing signals, K learnable latent queries aggregate $\mathbf{H}$ into compact representations $\mathbf{Z}^{(0)} \in \mathbb{R}^{K\times d}$. A latent Transformer then models cross-variable interactions within the compact latent space, yielding the interacted representations $\mathbf{Z} \in \mathbb{R}^{K\times d}$. Subsequently, cross-attention maps $\mathbf{Z}$ back to the original variable space and produces the recovered representations $\mathbf{U}^{\mathrm{rec}} \in \mathbb{R}^{C\times d}$. The main forecasting branch fuses $\mathbf{U}^{\mathrm{rec}}$ with the original temporal representations $\mathbf{H}$ to obtain $\mathbf{U}^{\mathrm{main}}$, from which the normalized forecasts $\widehat{\overline{\mathbf{Y}}}^{\mathrm{main}}$ are generated. Instance denormalization then transforms them into the final forecasts $\widehat{\mathbf{Y}}$. During training, the budget-conditioned preservation branch combines each recovered representation $\mathbf{u}_i^{\mathrm{rec}}$ with its budget embedding $\mathbf{e}_i^b$ to produce the auxiliary forecast $\widehat{\mathbf{y}}_i^{\mathrm{aux}}$. These auxiliary forecasts are aligned with the stop-gradient outputs $\widehat{\mathbf{y}}_i^{\mathrm{main}}$ of the main forecasting branch, encouraging the latent pathway to retain forecasting-relevant information under the allocated budgets. The preservation branch is removed during inference and introduces no additional inference cost.

3.2 Demand-Guided Latent Compression

Demand-guided latent compression consists of four components: 1) a demand analyzer, 2) an adaptive budget router, 3) demand-guided latent compression and interaction, and 4) variable-level recovery. The demand analyzer evaluates each variable through a shared compact bottleneck and uses

its reconstruction discrepancy as a comparable demand score. The adaptive budget router converts the resulting scores into continuous budget priors and routing signals. Guided by these signals, the compression module aggregates variable representations and models their interactions in the compact latent space. Finally, the recovery module maps the interacted latent representations back to the original variable space.

Demand Analyzer. To compare how adequately different variables can be represented under limited capacity, the demand analyzer evaluates them through the same temporal bottleneck. Sharing the bottleneck parameters and dimensionality establishes a common evaluation condition, making the resulting information loss comparable across variables. For each normalized variable sequence $\overline{\mathbf{x}}^{\,i} \in \mathbb{R}^L$, we compute:

$$\mathbf{z}_i^d = \phi_{\text{enc}}\left(\overline{\mathbf{x}}^{\,i}\right), \qquad \widetilde{\mathbf{x}}^{\,i} = \phi_{\text{dec}}\left(\mathbf{z}_i^d\right), \qquad d_i = \frac{1}{L}\left\|\overline{\mathbf{x}}^{\,i} - \widetilde{\mathbf{x}}^{\,i}\right\|_2^2, \tag{2}$$

where $\mathbf{z}_i^d \in \mathbb{R}^r$, $r < L$, and the encoder and decoder are shared across all variables. A larger d_i indicates that more temporal information cannot be recovered through the common bottleneck and therefore corresponds to a higher relative representational demand. The reconstruction objective of the demand analyzer is $\mathcal{L}_{\text{dem}} = \frac{1}{C}\sum_{i=1}^{C} d_i$.

Adaptive Budget Router. The raw demand scores do not directly specify how the limited latent capacity should be distributed, and their numerical scales may vary across input windows. The adaptive budget router therefore standardizes the scores within each input and transforms them into a constrained variable-wise allocation:

$$s_i = \frac{d_i - \mu_d}{\sigma_d + \epsilon}, \qquad \rho_i = \frac{\exp(s_i/\tau)}{\sum_{j=1}^{C}\exp(s_j/\tau)}, \qquad k_i = k_{\min} + \left(B_{\text{tot}} - Ck_{\min}\right)\rho_i, \tag{3}$$

where μ_d and σ_d are the mean and standard deviation of the demand scores, τ is a temperature parameter, B_{tot} denotes the total compression budget available for capacity allocation across variables, and $k_{\min}$ specifies the minimum capacity prior assigned to each variable. The allocation satisfies $\sum_{i=1}^{C}\rho_i = 1$ and $\sum_{i=1}^{C} k_i = B_{\text{tot}}$. The quantities s_i, ρ_i, and k_i are successive parameterizations of the same routing signal. Specifically, s_i represents the standardized relative demand, ρ_i defines the corresponding allocation distribution, and k_i expresses this distribution on an explicit continuous budget scale. The continuous prior k_i represents the intended relative capacity assigned to variable i, rather than a discrete number of latent queries. The standardized demand s_i is used for attention biasing, ρ_i for budget-usage alignment, and k_i for budget-conditioned prediction preservation.

Demand-Guided Latent Compression and Interaction. Budget priors describe the intended capacity allocation, but this allocation must be realized during latent aggregation. ComPass introduces K learnable latent queries $\mathbf{Q} \in \mathbb{R}^{K\times d}$, where $K \ll C$, to compress the variable representations. For the h-th attention head, the demand-biased attention matrix is:

$$\mathbf{A}_h = \text{Softmax}\left(\frac{\mathbf{Q}_h\mathbf{K}_h^\top}{\sqrt{d_h}} + \mathbf{1}_K\phi_{\text{bias}}(\mathbf{s})^\top\right) \in \mathbb{R}^{K\times C}, \tag{4}$$

where $\mathbf{s} = [s_1, \ldots, s_C]^\top$, $\mathbf{Q}_h$ and $\mathbf{K}_h$ are the query and key projections of $\mathbf{Q}$ and $\mathbf{H}$, respectively. The variable-wise bias injects the demand signal into the attention logits, allowing latent aggregation to adapt to heterogeneous representational demands. Applying $\mathbf{A}_h$ to the corresponding value projections and combining all attention heads produces the compact representation $\mathbf{Z}^{(0)} \in \mathbb{R}^{K\times d}$.

Attention biasing influences individual aggregation decisions but does not ensure that aggregate attention usage follows the intended budget allocation. We therefore measure the realized usage of each variable and align it with the routing distribution:

$$u_i = \frac{1}{mK}\sum_{h=1}^{m}\sum_{q=1}^{K}\left[\mathbf{A}_h\right]_{q,i}, \qquad \mathcal{L}_{\text{align}} = \sum_{i=1}^{C}\rho_i\log\left(\frac{\rho_i+\epsilon}{u_i+\epsilon}\right), \tag{5}$$

where m denotes the number of attention heads. Both $\mathbf{u} = [u_1, \ldots, u_C]^\top$ and $\boldsymbol{\rho} = [\rho_1, \ldots, \rho_C]^\top$ sum to one. Thus, demand-based biasing introduces the routing signal into individual attention logits, while budget-usage alignment regularizes the aggregate compression behavior according to the intended allocation. The compressed representations are then passed through a latent Transformer to model cross-variable interactions in the compact latent space, yielding $\mathbf{Z} = \mathcal{T}_{\text{lat}}(\mathbf{Z}^{(0)}) \in \mathbb{R}^{K\times d}$.

Variable-Level Recovery. After latent interaction, the compact representations are mapped back to the original variable space for forecasting. We perform this recovery through cross-attention:

$$\mathbf{U}^{\mathrm{rec}} = \mathrm{MHA}_{\mathrm{rec}}\left(\mathbf{H}, \mathbf{Z}, \mathbf{Z}\right) \in \mathbb{R}^{C\times d}. \tag{6}$$

where the original variable representations $\mathbf{H}$ serve as queries, while the interacted latent representations $\mathbf{Z}$ provide keys and values. This operation retrieves latent information according to each variable's temporal representation and produces variable-indexed recovered representations for subsequent forecasting.

3.3 Budget-Conditioned Prediction Preservation

Budget-conditioned prediction preservation consists of two components: 1) a main forecasting branch and 2) a budget-conditioned preservation branch. The main branch combines the original temporal representations with the information recovered from the latent space to produce the final forecasts. The preservation branch instead operates on the representations transmitted through the latent pathway and evaluates whether their information remains useful for forecasting under the allocated budgets.

Main Forecasting Branch. The final forecast should retain variable-specific temporal information while incorporating the cross-variable information recovered from the compact latent space. We therefore form the main representation through residual fusion, $\mathbf{U}^{\mathrm{main}} = \mathbf{H} + \mathbf{U}^{\mathrm{rec}} \in \mathbb{R}^{C\times d}$, and obtain the normalized main forecasts as $\widehat{\overline{\mathbf{Y}}}^{\mathrm{main}} = g_{\mathrm{main}}(\mathbf{U}^{\mathrm{main}}) \in \mathbb{R}^{S\times C}$. After inverse normalization, the main forecasts produce $\widehat{\mathbf{Y}} \in \mathbb{R}^{S\times C}$ and are supervised by the forecasting targets.

Budget-Conditioned Preservation Branch. To encourage the demand-guided latent pathway to preserve forecasting-relevant information acquired through compact cross-variable modeling, the preservation branch produces auxiliary forecasts directly from $\mathbf{U}^{\mathrm{rec}}$ and conditions them on the corresponding variable-wise budgets:

$$\mathbf{e}_i^b = \phi_{\mathrm{bud}}\left(\frac{k_i - \mu_k}{\sigma_k + \epsilon}\right), \qquad \widehat{\overline{\mathbf{y}}}_i^{\mathrm{aux}} = g_{\mathrm{aux}}\left(\left[\mathbf{u}_i^{\mathrm{rec}} \,\|\, \mathbf{e}_i^b\right]\right), \tag{7}$$

where μ_k and σ_k are the mean and standard deviation of the variable-wise budgets, and $\|$ denotes concatenation. The shared mapping $\phi_{\mathrm{bud}} : \mathbb{R} \to \mathbb{R}^d$ transforms the standardized budget k_i into the budget embedding $\mathbf{e}_i^b$. The auxiliary head g_{aux} then combines $\mathbf{e}_i^b$ with the recovered representation $\mathbf{u}_i^{\mathrm{rec}}$ to produce the normalized S-step forecast $\widehat{\overline{\mathbf{y}}}_i^{\mathrm{aux}}$ for variable i. The resulting auxiliary forecasts are aligned with the stop-gradient outputs of the target-supervised main forecasting branch:

$$\mathcal{L}_{\mathrm{pres}} = \frac{1}{CS}\sum_{i=1}^{C}\left\|\widehat{\overline{\mathbf{y}}}_i^{\mathrm{aux}} - \mathrm{sg}\left(\widehat{\overline{\mathbf{y}}}_i^{\mathrm{main}}\right)\right\|_2^2. \tag{8}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. The main forecasts combine the variable representations $\mathbf{H}$ with the recovered representations $\mathbf{U}^{\mathrm{rec}}$ and are learned from forecasting targets, thereby providing a task reference for the preservation branch. Minimizing $\mathcal{L}_{\mathrm{pres}}$ regularizes the compression and recovery pathway to retain predictive information under the allocated budgets. The auxiliary branch is used only during training and introduces no inference cost.

3.4 Training Objective

We use the mean squared error as the forecasting loss: $\mathcal{L}_{\mathrm{fore}} = \frac{1}{CS}\|\mathbf{Y} - \widehat{\mathbf{Y}}\|_F^2$. To reduce redundancy among latent queries, we further introduce a diversity loss $\mathcal{L}_{\mathrm{div}} = \|\overline{\mathbf{A}}^{n}(\overline{\mathbf{A}}^{n})^{\top} - \mathbf{I}_K\|_F^2$, where $\overline{\mathbf{A}}^{n}$ denotes the row-normalized, head-averaged compression attention map. We group the demand analysis, budget–usage alignment, and latent diversity objectives into the demand-guided compression loss $\mathcal{L}_{\mathrm{dgc}} = \mathcal{L}_{\mathrm{dem}} + \mathcal{L}_{\mathrm{align}} + \mathcal{L}_{\mathrm{div}}$. The overall training objective is:

$$\mathcal{L} = \mathcal{L}_{\mathrm{fore}} + \mathcal{L}_{\mathrm{dgc}} + \mathcal{L}_{\mathrm{pres}}. \tag{9}$$

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate ComPass on 16 high-dimensional multivariate time series datasets (Ni et al., 2025): *Neurolib*, *Solar*, *Atec*, *Meters*, *Temp*, *Wind*, *Traffic-CA*, *Traffic-GLA*, *Traffic-GBA*, *Air Quality*, *SIRS*, *SP500*, *M5*, *Measles*, *Wiki-20k*, and *Mobility*. These datasets span diverse domains and contain between 1,105 and 20,000 variables, with sampling frequencies ranging from milliseconds to days. Detailed descriptions, statistics, and prediction horizons are provided in Appendix A.1.

Baselines. We compare ComPass with nine representative forecasting methods from three categories: 1) channel-independent models, including DLinear (Zeng et al., 2023), PAttn (Tan et al., 2024), and PatchTST (Nie et al., 2022); 2) channel-dependent models, including iTransformer (Liu et al., 2023), TSMixer (Chen et al., 2023), TimesNet (Wu et al., 2023a), CCM (Chen et al., 2024), and DUET (Qiu et al., 2025b); and 3) the high-dimensional forecasting model U-CAST (Ni et al., 2025). Further details are provided in Appendix A.2.

Metrics. Following previous forecasting studies, we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics, where lower values indicate better forecasting performance. Their definitions are provided in Appendix A.3.

Implementation Details. All methods use the same data splits and evaluation protocol. Baselines follow the recommended configurations from their original papers or official implementations. Experiments are implemented in PyTorch and conducted on an NVIDIA Tesla-A800 GPU. Additional training and implementation details are provided in Appendix A.4.

Table 1: Full forecasting results across models and datasets. Prediction length S is provided in Table 4. Lower values indicate better forecasting performance. Red: best; blue: second best. Ties are counted for all tied methods.

Models	**Channel-Independent**						**Channel-Dependent**										**High-Dimensional Forecasting**			
	DLinear		PAttn		PatchTST		iTransformer		TSMixer		TimesNet		CCM		DUET		U-CAST		ComPass	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Atec	0.318	0.314	0.299	0.275	0.298	0.298	0.345	0.319	0.398	0.387	0.493	0.429	0.362	0.346	0.330	0.339	0.287	0.280	**0.283**	**0.272**
Air Quality	0.449	0.446	0.449	0.432	0.448	0.432	0.447	0.431	0.447	0.438	0.457	0.438	0.479	0.491	0.452	0.444	0.446	0.430	**0.423**	**0.415**
Temp	0.272	0.391	0.278	0.395	0.279	0.396	0.265	0.389	0.266	0.389	0.287	0.408	0.379	0.479	0.435	0.511	0.262	**0.383**	**0.262**	0.384
Wind	1.128	0.697	1.256	0.758	1.254	0.757	1.116	0.699	1.346	0.742	1.161	0.708	1.165	0.698	1.227	0.746	1.104	0.697	**1.076**	**0.686**
Mobility	0.344	0.359	0.337	0.336	0.344	0.341	0.312	0.314	1.165	0.787	0.410	0.388	0.523	0.468	0.439	0.410	0.315	0.317	**0.304**	**0.309**
Traffic-CA	0.063	0.141	0.491	0.554	0.295	0.417	0.271	0.391	0.082	0.186	0.101	0.205	0.067	0.153	—	—	0.061	0.131	**0.056**	**0.127**
Traffic-GBA	0.062	0.137	0.059	0.132	0.060	0.139	0.063	0.141	0.074	0.168	0.098	0.204	0.067	0.151	0.650	0.638	0.059	0.126	**0.054**	**0.122**
Traffic-GLA	0.062	0.142	0.060	0.136	0.060	0.136	0.065	0.145	0.071	0.168	0.094	0.199	0.066	0.152	0.810	0.738	0.060	0.132	**0.056**	**0.129**
M5	3.688	0.870	3.650	0.867	3.655	0.872	3.549	0.853	6.863	1.623	4.490	0.919	4.916	0.941	3.768	0.880	3.501	0.849	**3.394**	**0.832**
Measles	0.128	0.252	0.011	0.048	0.013	0.058	0.010	0.048	0.560	0.549	0.018	0.060	0.244	0.323	0.015	0.064	**0.010**	0.042	0.011	**0.040**
Neurolib	1.793	0.381	2.458	0.445	2.395	0.438	1.718	0.347	2.240	0.532	2.475	0.458	1.774	0.403	2.519	0.451	1.750	0.350	**1.667**	**0.343**
Solar	0.174	0.255	0.604	0.582	0.416	0.469	0.343	0.427	**0.155**	**0.216**	0.157	0.224	0.177	0.258	—	—	0.172	0.246	0.158	0.219
SIRS	0.058	0.168	0.025	0.109	0.033	0.129	0.028	0.113	0.016	0.078	0.162	0.327	0.048	0.156	0.095	0.236	0.007	0.052	**0.003**	**0.032**
Meters	0.944	0.549	0.941	0.552	1.254	0.706	0.949	0.556	0.987	0.564	1.034	0.586	0.946	0.551	1.308	0.731	0.943	0.551	**0.937**	**0.549**
SP500	0.630	0.367	0.516	0.309	0.523	0.313	0.511	0.306	2.674	1.120	0.611	0.343	0.727	0.414	0.568	0.335	0.502	0.301	**0.456**	**0.282**
Wiki-20k	10.740	0.394	10.290	0.306	10.291	0.305	10.933	0.405	10.446	0.332	10.586	0.325	11.413	0.373	10.278	0.304	10.273	0.302	**10.249**	**0.297**
1st Count	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	**14**	**14**

4.2 MAIN RESULTS

Table 1 reports the forecasting performance of ComPass and all baselines on 16 high-dimensional datasets. ComPass achieves the best MSE and MAE on 14 of the 16 datasets, demonstrating strong performance across diverse domains and variable dimensions. Compared with the optimal prediction baseline, ComPass obtains lower forecasting errors on most datasets, with particularly clear improvements on *Neurolib*, *Air Quality*, *SIRS*, *SP500*, and the three traffic datasets.

Channel-independent models remain competitive on several datasets because of their robust and scalable temporal modeling, but cannot explicitly exploit cross-variable dependencies. Channel-dependent models capture such dependencies, yet their performance varies considerably across high-dimensional datasets, indicating that introducing channel interactions alone does not ensure reliable forecasting at scale. By adapting latent compression to heterogeneous variable demands and preserving predictive information under the allocated budgets, ComPass achieves more consistent performance across different datasets.

4.3 Model Analysis

We conduct ablation studies to evaluate the demand analyzer, adaptive budget router, demand-guided latent compression, budget-conditioned prediction preservation, and auxiliary training objectives. We conducted additional analytical experiments, which are presented in Appendix B.

Ablation study. To evaluate the contribution of each component in ComPass, we consider the following variants: (a) removing the shared bottleneck demand analyzer; (b) replacing adaptive budgets with uniform allocation; (c) replacing demand-based budgets with random allocation; (d) replacing demand-guided latent attention with linear compression; (e) removing the preservation head; (f) removing the budget embedding from the preservation branch; (g) removing the budget–usage alignment loss $\mathcal{L}_{\text{align}}$; and (h) removing the latent diversity loss $\mathcal{L}_{\text{div}}$. Table 2 reports the results on three representative high-dimensional datasets.

We make the following observations. Firstly, the performance significantly deteriorated after removing the requirement analyzer. Uniform and random budgets also perform consistently worse than ComPass, showing that nonuniformity alone is insufficient and that capacity allocation should reflect the relative representational demands estimated under the shared bottleneck.

Second, replacing latent attention with linear compression causes substantial performance degradation, particularly on Neurolib. Linear compression uses fixed variable mixing and cannot adapt latent aggregation to the input-dependent routing signal. In contrast, demand-guided latent attention allows the compact queries to aggregate variable information according to the estimated allocation.

Table 2: Ablation results on representative datasets. Each entry reports MSE/MAE.

Variation	Neurolib	SP500	Air Quality
w/o Demand Analyzer	1.850/0.371	0.502/0.304	0.460/0.438
Uniform Budget	1.731/0.356	0.486/0.295	0.447/0.429
Random Budget	1.736/0.353	0.494/0.299	0.452/0.433
Linear Compression	1.846/0.383	0.490/0.298	0.455/0.435
w/o Preservation Head	1.727/0.352	0.464/0.286	0.430/0.420
w/o Budget Embedding	1.703/0.353	0.472/0.290	0.436/0.424
w/o $\mathcal{L}_{\text{align}}$	1.743/0.361	0.476/0.292	0.439/0.425
w/o $\mathcal{L}_{\text{div}}$	1.779/0.368	0.466/0.287	0.432/0.421
ComPass	**1.667/0.343**	**0.456/0.282**	**0.423/0.415**

Third, removing either the preservation head or its budget embedding consistently reduces forecasting accuracy. The former eliminates prediction-level supervision on the recovered representations, while the latter turns the preservation branch into a generic auxiliary predictor without explicit allocation context. These results show that budget-conditioned prediction consistency helps the latent pathway retain forecasting-relevant information under variable-wise budgets.

Finally, removing either $\mathcal{L}_{\text{align}}$ or $\mathcal{L}_{\text{div}}$ leads to performance drops. The alignment loss encourages realized attention usage to follow the intended allocation, whereas the diversity loss reduces redundant aggregation among latent queries. Their complementary effects improve the coverage and effectiveness of compact latent representations.

4.4 Efficiency and Scalability

Beyond forecasting accuracy, high-dimensional forecasting also requires manageable computational costs as the number of variables increases. We evaluate ComPass from two perspectives: its accuracy-efficiency trade-off on Wiki-20k and its scalability under increasing variable dimensions.

Memory Usage and Runtime Efficiency. We compare ComPass with representative forecasting models on Wiki-20k, which contains 20,000 variables. All models are evaluated under the same hardware and evaluation protocol. Fig. 3 reports the forecasting error, inference latency per sample, and peak GPU memory usage.

ComPass achieves the lowest forecasting error while maintaining practical inference efficiency. Compared with U-CAST, the most directly related high-dimensional baseline, ComPass reduces inference latency from 9.81 ms to 8.23 ms per sample, corresponding to a 16.1% reduction, while obtaining lower forecasting error. PatchTST, PAttn, and iTransformer exhibit substantially higher latency at this variable scale, while PAttn and iTransformer also require larger memory footprints. Although TSMixer has relatively low computational costs, its forecasting error is considerably higher.

This favorable trade-off benefits from modeling variable interactions within a compact latent space rather than constructing dense pairwise interactions among all variables. The demand analyzer and budget router provide lightweight signals for guiding latent aggregation, while the budget-conditioned preservation branch is removed during inference. Consequently, ComPass improves the forecasting utility of latent compression without introducing additional inference cost.

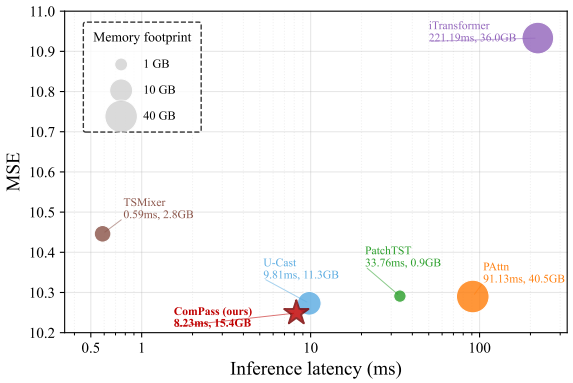
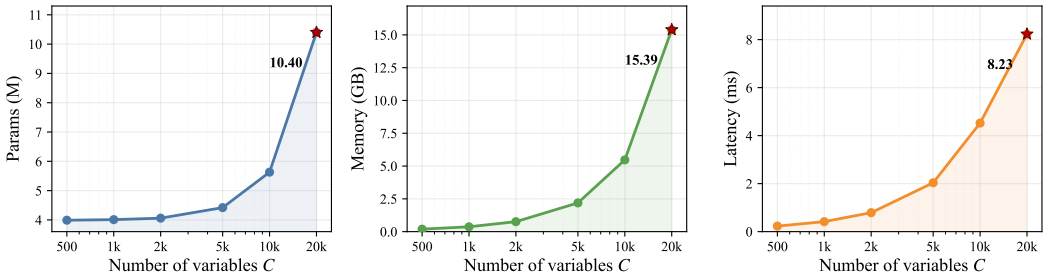


Figure 3: Accuracy–efficiency comparison on Wiki-20k ($C = 20{,}000$). Latency is measured per sample, and marker size denotes peak GPU memory.

Scalability Study. We further examine how the computational cost of ComPass changes with the variable dimension. We construct subsets of Wiki-20k containing between 500 and 20,000 variables while keeping the remaining settings unchanged. Fig. 4 reports peak GPU memory, inference latency, and model parameters at different scales.

Memory usage and inference latency increase steadily with the number of variables because more variable representations must be encoded, compressed, and recovered. Nevertheless, ComPass remains computationally feasible at the largest scale, requiring only 8.23 ms of inference latency per sample at 20,000 variables. Moreover, when the variable dimension increases by $40\times$, from 500 to 20,000, the parameter count increases by approximately $2.61\times$, from 3.99M to 10.40M. These results demonstrate that ComPass maintains manageable memory, latency, and parameter costs as the variable dimension grows.



(a) Model parameters (b) GPU memory (c) Inference latency

Figure 4: Scalability of ComPass on Wiki-20k with respect to the number of variables. Peak GPU memory, inference latency per sample, and model parameters are reported at different variable dimensions.

5 Conclusion

In this work, we introduced ComPass, a demand-guided latent compression framework for high-dimensional time series forecasting. ComPass replaces uniform compression with differentiated capacity allocation by translating heterogeneous variable demands into continuous budget priors. ComPass further introduces a budget-conditioned prediction preservation objective that evaluates recovered representations by their forecasting utility, encouraging information useful for forecasting to remain available after compression and recovery. Extensive experiments on 16 high-dimensional forecasting benchmarks demonstrate that ComPass achieves competitive or superior performance while scaling to time series with up to 20,000 variables. Further analyses validate the effectiveness of demand-guided capacity allocation and budget-conditioned prediction preservation. These results highlight its practical potential for large-scale forecasting applications.

AI USE STATEMENT

We do not use generative AI tools in the preparation of this work. The research ideas, model implementation, experiments, and language editing are developed and verified by the authors.

ETHICS STATEMENT

Our work exclusively uses publicly available benchmark datasets that contain no personally identifiable information. No human subjects are involved in this research.

REPRODUCIBILITY STATEMENT

The performance of ComPass and the datasets used in our work are real, and all experimental results can be reproduced. We have released our model introduction in an anonymous repository: https://anonymous.4open.science/r/ComPass–BD5C. Once the paper is accepted, we will re-release the complete model code and parameters.

REFERENCES

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv*, 2018.

David Campos, Tung Kieu, Chenjuan Guo, Feiteng Huang, Kai Zheng, Bin Yang, and Christian S Jensen. Unsupervised time series outlier detection with diversity-driven convolutional ensembles. *Proceedings of the VLDB Endowment*, 2021.

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. In *Advances in Neural Information Processing Systems (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. *arXiv*, 2023.

Razvan-Gabriel Cirstea, Bin Yang, Chenjuan Guo, Tung Kieu, and Shirui Pan. Towards spatio-temporal aware traffic time series forecasting. In *IEEE International Conference on Data Engineering (ICDE)*, 2022.

Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Jigang Bao, Yong Jiang, and Shu-Tao Xia. Periodicity decoupling framework for long-term series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024.

Ziquan Fang, Lu Pan, Lu Chen, Yuntao Du, and Yunjun Gao. MDTP: A multi-source deep traffic prediction framework over spatio-temporal trajectory data. *Proceedings of the VLDB Endowment*, 2021.

Xingtong Ge, Jixiang Luo, Xinjie Zhang, Tongda Xu, Guo Lu, Dailan He, Jing Geng, Yan Wang, Jun Zhang, and Hongwei Qin. Task-aware encoder control for deep video compression. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Clive William John Granger and Paul Newbold. *Forecasting Economic Time Series*. Academic Press, 2014.

Shiyan Hu, Jianxin Jin, Yang Shu, Peng Chen, Bin Yang, and Chenjuan Guo. Towards multimodal time series anomaly detection with semantic alignment and condensed interaction. In *International Conference on Learning Representations (ICLR)*, 2026a.

Shiyan Hu, Kai Zhao, Chenjuan Guo, Xiangfei Qiu, Yang Shu, Jilin Hu, Christian S. Jensen, and Bin Yang. TAP: Time series anomaly prediction via adaptive period modeling and dual representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 2026b.

Yifan Hu, Guibin Zhang, Peiyuan Liu, Disen Lan, Naiqi Li, Dawei Cheng, Tao Dai, Shu-Tao Xia, and Shirui Pan. Timefilter: Patch-specific spatial-temporal graph filtration for time series forecasting. In *International Conference on Machine Learning (ICML)*, 2025.

Qihe Huang, Lei Shen, Ruixin Zhang, Shouhong Ding, Binwu Wang, Zhengyang Zhou, and Yang Wang. CrossGNN: Confronting noisy multivariate time series via cross interaction refinement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Duc Kieu, Tung Kieu, Peng Han, Bin Yang, Christian S. Jensen, and Bac Le. TEAM: Topological evolution-aware framework for traffic forecasting. *Proceedings of the VLDB Endowment*, 2025.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2018.

Zhe Li, Xiangfei Qiu, Peng Chen, Yihang Wang, Hanyin Cheng, Yang Shu, Jilin Hu, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, and Bin Yang. TSFM-Bench: A comprehensive and unified benchmarking of foundation models for time series forecasting. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2025.

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 (ICLR)*, 2023.

Juntong Ni, Shiyu Wang, Zewen Liu, Xiaoming Shi, Xinyue Zhong, Zhou Ye, and Wei Jin. U-CAST: Learning hierarchical structures for high-dimensional time series forecasting. *arXiv*, 2025.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

Unki Park, Seongmoon Jeong, Youngchan Jang, Gyeong-Moon Park, and Jong Hwan Ko. Test-time fine-tuning of image compression models for multi-task adaptability. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. TFB: Towards comprehensive and fair benchmarking of time series forecasting methods. *Proceedings of the VLDB Endowment*, 2024.

Xiangfei Qiu, Zhe Li, Wanghui Qiu, Shiyan Hu, Lekui Zhou, Xingjian Wu, Zhengyu Li, Chenjuan Guo, Aoying Zhou, Zhenli Sheng, Jilin Hu, Christian S. Jensen, and Bin Yang. TAB: Unified benchmarking of time series anomaly detection methods. *Proceedings of the VLDB Endowment*, 2025a.

Xiangfei Qiu, Xingjian Wu, Yan Lin, Chenjuan Guo, Jilin Hu, and Bin Yang. DUET: Dual clustering enhanced multivariate time series forecasting. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2025b.

Mingtian Tan, Mike Merrill, Vinayak Gupta, Tim Althoff, and Tom Hartvigsen. Are language models actually useful for time series forecasting? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Meng Wang, Jintao Yang, Bin Yang, Hui Li, Tongxin Gong, Bo Yang, and Jiangtao Cui. Towards lightweight time series forecasting: A patch-wise transformer with weak data enriching. In *IEEE International Conference on Data Engineering (ICDE)*, 2025.

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 (ICLR)*, 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.

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 (ICLR)*, 2023a.

Haixu Wu, Hang Zhou, Mingsheng Long, and Jianmin Wang. Interpretable weather forecasting for worldwide stations with a unified deep model. *Nature Machine Intelligence*, 2023b.

Xingjian Wu, Xiangfei Qiu, Zhengyu Li, Yihang Wang, Jilin Hu, Chenjuan Guo, Hui Xiong, and Bin Yang. CATCH: Channel-aware multivariate time series anomaly detection via frequency patching. In *International Conference on Learning Representations (ICLR)*, 2025a.

Xinle Wu, Xingjian Wu, Bin Yang, Lekui Zhou, Chenjuan Guo, Xiangfei Qiu, Jilin Hu, Zhenli Sheng, and Christian S Jensen. AutoCTS++: Zero-shot joint neural architecture and hyperparameter search for correlated time series forecasting. *The VLDB Journal*, 2024.

Xinle Wu, Xingjian Wu, Dalin Zhang, Miao Zhang, Chenjuan Guo, Bin Yang, and Christian S Jensen. Fully automated correlated time series forecasting in minutes. *Proceedings of the VLDB Endowment*, 2025b.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2020.

Hengyu Ye, Jiadong Chen, Fuxin Jiang, Xiao He, Tieying Zhang, Jianjun Chen, and Xiaofeng Gao. Fremer: Lightweight and effective frequency transformer for workload forecasting in cloud services. *Proceedings of the VLDB Endowment*, 2025.

Xueyan Yin, Genze Wu, Jinze Wei, Yanming Shen, Heng Qi, and Baocai Yin. Deep learning on traffic prediction: Methods, analysis, and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2023.

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 *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning (ICML)*, 2022.

A Experimental Details

A.1 Datasets

We evaluate ComPass on 16 high-dimensional multivariate time series datasets introduced in the U-CAST benchmark (Ni et al., 2025). These datasets cover diverse application domains, including neuroscience, energy, cloud computing, weather, traffic, environmental monitoring, epidemiology, finance, retail, web traffic, and social mobility. Their variable dimensions range from 1,105 to 20,000, while their sampling frequencies range from milliseconds to days.

Table 3 summarizes several conventional forecasting benchmarks for comparison. These datasets typically contain fewer than one thousand variables, with their dimensions ranging from 7 to 862. In contrast, all datasets used in our experiments contain more than one thousand variables, making them more suitable for evaluating forecasting models under high-dimensional settings.

Table 3: Statistics of conventional time series forecasting datasets.

Dataset	Domain	Frequency	Variables
ETTh1	Electricity	1 hour	7
ETTh2	Electricity	1 hour	7
ETTm1	Electricity	15 mins	7
ETTm2	Electricity	15 mins	7
Weather	Environment	10 mins	21
Exchange	Economics	1 day	8
Electricity	Electricity	1 hour	321
Traffic	Traffic	1 hour	862

Table 4 reports the statistics of the 16 high-dimensional datasets. The datasets contain between 974 and 105,120 timestamps and cover variable dimensions from approximately one thousand to twenty thousand. The prediction length is configured according to the sampling frequency of each dataset. Specifically, we use a prediction length of 336 for *Neurolib*, *Solar*, *Atec*, and *Meters*; 168 for the hourly *Temp*, *Wind*, and traffic datasets; 28 for the six-hourly *Air Quality* dataset; and 7 for the daily datasets. This configuration provides forecasting horizons appropriate for their respective temporal resolutions.

Table 4: Statistics of the high-dimensional forecasting datasets used in our experiments.

Dataset	Variables	Timestamps	Frequency	Pred. Len.	Storage	Domain
Neurolib	2,000	60,000	1 ms	336	2.44 GB	Neuroscience
Solar	5,162	105,120	5 mins	336	2.33 GB	Energy
Atec	1,569	8,928	10 mins	336	158.8 MB	Cloud Computing
Meters	2,898	28,512	30 mins	336	651 MB	Energy
Temp	3,850	17,544	1 hour	168	383.6 MB	Weather
Wind	3,850	17,544	1 hour	168	331.4 MB	Weather
Traffic-CA	7,491	43,824	1 hour	168	2.48 GB	Traffic
Traffic-GLA	3,376	43,824	1 hour	168	1.12 GB	Traffic
Traffic-GBA	2,229	43,824	1 hour	168	689.1 MB	Traffic
Air Quality	1,105	15,461	6 hours	28	94.6 MB	Environment
SIRS	2,994	9,000	1 day	7	675 MB	Epidemiology
SP500	1,475	7,553	1 day	7	184.1 MB	Finance
M5	3,049	1,941	1 day	7	13.7 MB	Retail
Measles	1,161	1,330	1 day	7	12.1 MB	Epidemiology
Wiki-20k	20,000	2,557	1 day	7	289 MB	Web
Mobility	5,826	974	1 day	7	29.7 MB	Social Mobility

A.2 Baselines

We compare ComPass with nine representative forecasting models covering channel-independent modeling, channel-dependent modeling, and latent compression for high-dimensional forecasting.

Channel-Independent Models. These methods share temporal modeling parameters across variables while processing each variable independently during the forward pass. They generally scale well with the variable dimension but do not explicitly model cross-variable interactions.

- **DLinear** (Zeng et al., 2023) decomposes each time series into trend and remainder components and applies linear projections to produce forecasts. Its simple channel-independent architecture provides a strong and efficient baseline.
- **PAttn** (Tan et al., 2024) applies patch-based attention to capture temporal patterns while retaining channel-independent processing, providing an efficient attention-based forecasting baseline.
- **PatchTST** (Nie et al., 2022) divides each univariate series into temporal patches and applies a shared Transformer encoder under the channel-independent setting.

Channel-Dependent Models. These methods explicitly model dependencies among variables through attention, feature mixing, convolution, or channel clustering.

- **iTransformer** (Liu et al., 2023) represents each variable as a token and applies self-attention across variable tokens to capture multivariate dependencies.
- **TSMixer** (Chen et al., 2023) employs multilayer perceptrons to mix information along the temporal and feature dimensions, providing an all-MLP architecture for multivariate forecasting.
- **TimesNet** (Wu et al., 2023a) transforms one-dimensional time series into two-dimensional representations according to their dominant periods and uses two-dimensional convolutions to model temporal variations.
- **CCM** (Chen et al., 2024) organizes variables according to their learned similarities and performs dependency modeling within channel clusters, reducing unnecessary interactions between weakly related variables.
- **DUET** (Qiu et al., 2025b) employs dual clustering to organize temporal patterns and channel relationships, enabling selective dependency modeling among related variables.

High-Dimensional Forecasting Model. We further compare ComPass with U-CAST, which is specifically designed for forecasting time series with very large variable dimensions.

- **U-CAST** (Ni et al., 2025) learns hierarchical structures for high-dimensional forecasting and uses latent representations to reduce the cost of modeling large numbers of variables. It serves as the most closely related baseline for evaluating the effectiveness of demand-guided latent compression.

A.3 Evaluation Metrics

Following previous time series forecasting studies, we evaluate all methods using Mean Squared Error (MSE) and Mean Absolute Error (MAE). Given the ground-truth future sequence $\mathbf{Y} \in \mathbb{R}^{S\times C}$ and the corresponding prediction $\widehat{\mathbf{Y}} \in \mathbb{R}^{S\times C}$, MSE is defined as:

$$\mathrm{MSE} = \frac{1}{CS}\sum_{i=1}^{C}\sum_{j=1}^{S}\left(Y_j^i - \widehat{Y}_j^i\right)^2, \tag{10}$$

and MAE is defined as:

$$\mathrm{MAE} = \frac{1}{CS}\sum_{i=1}^{C}\sum_{j=1}^{S}\left|Y_j^i - \widehat{Y}_j^i\right|. \tag{11}$$

Both metrics measure the discrepancy between the predicted and ground-truth future sequences, and lower values indicate better forecasting performance.

A.4 Implementation Details

All experiments are implemented in PyTorch (Paszke et al., 2019) with Python 3.10 and conducted on an NVIDIA Tesla-A800 GPU. For each dataset, the look-back length is configured according to its prediction horizon. The same input and prediction lengths, data splits, and evaluation protocol are used for ComPass and all baselines on each dataset.

ComPass is optimized using Adam with an initial learning rate of 5×10^{-4}. StepLR is applied for learning-rate decay during training, and early stopping with a patience of five epochs is used to prevent overfitting. The initial batch size is set to 32. If an out-of-memory error occurs, the batch size is halved until the model can be trained. For all baseline methods, we use their official or publicly available implementations and follow the configurations recommended in their original papers whenever available. All methods are trained and evaluated on the same hardware.

Following recent studies on reliable time series evaluation (Qiu et al., 2024; 2025a; Li et al., 2025), we disable the "drop last" operation during testing. When `drop_last=True`, incomplete final batches are discarded, causing the evaluated test samples and reported results to vary with the test batch size. We therefore set `drop_last=False` for all methods to ensure that every test sample is included and that the comparison follows a consistent evaluation protocol.

B ADDITIONAL EXPERIMENTAL RESULTS

B.1 PARAMETER SENSITIVITY

We study two key hyperparameters of ComPass: the compression factor γ, which determines the number of latent queries through $K = \lceil C/\gamma \rceil$, and the bottleneck dimension r of the demand analyzer. Results on Neurolib, Wind, and Wiki-20k are reported in Tables 5 and 6.

Compression factor γ. The compression factor controls the overall capacity of the compact latent space. A smaller γ produces more latent queries and therefore applies lighter compression at a higher computational cost, whereas a larger γ imposes stronger compression. We evaluate $\gamma \in \{8, 16, 32, 64\}$ while keeping the remaining settings fixed. Importantly, γ determines the total latent capacity, while the demand-guided routing mechanism controls how this capacity is used according to variable-wise representational demands.

As shown in Table 5, $\gamma = 16$ achieves the best results on Neurolib and Wind, whereas Wiki-20k benefits from the larger latent capacity provided by $\gamma = 8$. This variation reflects differences in variable redundancy and interaction structure across datasets. Rather than tuning the compression factor separately for each dataset, we adopt $\gamma = 16$ as the default because it provides a favorable overall balance between forecasting accuracy and computational efficiency.

Table 5: Sensitivity to the compression factor γ, where $K = \lceil C/\gamma \rceil$. Lower values are better.

γ	Neurolib		Wind		Wiki-20k	
	MSE	MAE	MSE	MAE	MSE	MAE
8	2.458	0.452	1.078	0.687	**10.136**	**0.291**
16	**1.667**	**0.343**	**1.076**	**0.686**	10.249	0.297
32	1.773	0.368	1.079	0.688	10.224	0.298
64	1.702	0.359	1.082	0.690	10.164	0.294

Bottleneck dimension r. The bottleneck dimension r controls the capacity of the shared reconstruction bottleneck used to estimate variable demands. An excessively small r may cause reconstruction errors to be dominated by severe compression rather than differences in variable recoverability. Conversely, an overly large r makes reconstruction easier and may reduce the distinction between variables with different representational demands. We evaluate $r \in \{2, 4, 8, 16\}$ while keeping all other settings fixed.

As reported in Table 6, no single value dominates every dataset and metric. In particular, $r = 8$ achieves the lowest MSE on Neurolib and the best results on Wind, while $r = 16$ performs best on Wiki-20k and $r = 2$ gives the lowest Neurolib MAE. We therefore use $r = 8$ as the default because it provides consistently competitive performance across the evaluated datasets without dataset-specific tuning.

Table 6: Sensitivity to the bottleneck dimension r of the demand analyzer. Lower values are better.

r	Neurolib		Wind		Wiki-20k	
	MSE	MAE	MSE	MAE	MSE	MAE
2	1.669	**0.337**	1.080	0.688	10.244	0.296
4	1.784	0.368	1.081	0.689	10.251	0.300
8	**1.667**	0.343	**1.076**	**0.686**	10.249	0.297
16	1.738	0.355	1.078	0.687	**10.242**	**0.292**

B.2 COMPARISON WITH ALTERNATIVE COMPRESSION STRATEGIES

We further compare ComPass with two alternative compression strategies to examine whether generic variable compression is sufficient for high-dimensional forecasting. **Pooling Compression** aggregates variable representations through group-wise averaging, while **Similarity Compression** groups and aggregates variables according to their representation similarity. All variants are eval-

uated under the same experimental protocol. The results on Neurolib, Air Quality, and SP500 are reported in Table 7.

Table 7: Comparison of different compression strategies. Lower values are better.

Strategy	Neurolib		Air Quality		SP500	
	MSE	MAE	MSE	MAE	MSE	MAE
Pooling	1.905	0.385	0.474	0.449	0.548	0.326
Similarity	1.790	0.361	0.454	0.435	0.514	0.309
ComPass	**1.667**	**0.343**	**0.423**	**0.415**	**0.456**	**0.282**

Pooling Compression produces the largest forecasting errors across all three datasets, indicating that fixed group-wise aggregation may discard informative variable patterns during compression. Similarity Compression consistently improves upon pooling, suggesting that representation similarity provides useful guidance for variable aggregation. Nevertheless, similarity does not directly characterize how adequately each variable can be represented under limited latent capacity and therefore cannot explicitly account for heterogeneous representational demands.

ComPass achieves the lowest MSE and MAE on all three datasets. Compared with Similarity Compression, it reduces MSE by approximately 6.9% on Neurolib, 6.8% on Air Quality, and 11.3% on SP500. These results demonstrate that the performance gain of ComPass does not come solely from reducing the variable dimension. Instead, estimating variable-wise demands and using the resulting routing signals to guide latent aggregation enables the limited capacity to be used more effectively.

B.3 Scalability under Different Latent-Query Policies

By default, ComPass sets $K = \lceil C/\gamma \rceil$ to maintain a consistent compression ratio across datasets without dataset-specific tuning. To investigate whether ComPass supports stronger scalability, we additionally evaluate two sub-linear policies, $K = \lceil 2\sqrt{C} \rceil$ and $K = \lceil 8 \log C \rceil$, and three fixed settings, $K \in \{128, 256, 512\}$. All other configurations remain unchanged.

Table 8: Forecasting performance under different latent-query scaling policies. Lower values are better.

K Policy	Neurolib		Wind		Traffic-CA		Wiki-20k	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$K = \lceil C/\gamma \rceil$	1.667	0.343	**1.076**	**0.686**	**0.056**	**0.127**	**10.249**	**0.297**
$K = \lceil 2\sqrt{C} \rceil$	1.686	0.350	1.083	0.689	0.057	0.129	10.272	0.299
$K = \lceil 8 \log C \rceil$	1.725	0.360	1.094	0.694	0.060	0.133	10.338	0.305
$K = 128$	1.668	0.344	1.082	0.689	0.058	0.131	10.312	0.303
$K = 256$	**1.664**	**0.342**	1.077	**0.686**	0.057	0.129	10.274	0.299
$K = 512$	1.669	0.346	1.079	0.687	**0.056**	0.128	10.257	0.298

As shown in Table 8, the default proportional policy achieves the most consistently strong performance across the evaluated datasets. More importantly, ComPass remains competitive when the number of latent queries grows sub-linearly with C or is fixed across datasets. In particular, $K = 256$ and $K = 512$ produce results close to the default setting on all four datasets, indicating that demand-guided routing can effectively use a substantially smaller latent budget.

On Wiki-20k with $C = 20{,}000$, setting $K = 256$ and $K = 512$ reduces peak GPU memory from 15.39 GB to 5.10 GB and 7.80 GB, respectively, and reduces inference latency from 8.23 ms to 4.70 ms and 5.60 ms. These results show that the effectiveness of ComPass does not depend on letting K grow linearly with the variable dimension. We retain $K = \lceil C/\gamma \rceil$ in the main experiments because it maintains a consistent compression ratio without dataset-specific tuning and provides the most reliable overall forecasting accuracy.

B.4 VARIABLE DEMAND VISUALIZATION

To examine what is captured by the shared bottleneck demand analyzer, we visualize representative normalized variables $\overline{\mathbf{x}}^i$ associated with low and high demand scores d_i. As shown in Fig. 5, low-demand variables generally exhibit smoother and more regular temporal patterns, whereas high-demand variables tend to contain stronger fluctuations or temporal changes. Under the same compact bottleneck, the latter patterns are more difficult to reconstruct and therefore produce larger reconstruction errors.

This visualization provides an intuitive interpretation of the demand scores rather than defining demand according to temporal irregularity. Specifically, d_i quantifies the relative reconstruction loss of variable i under the same shared bottleneck. The visual patterns are therefore observations associated with different demand levels rather than predefined criteria for computing demand. These results indicate that the demand analyzer captures differences in bottleneck recoverability that can guide latent compression.

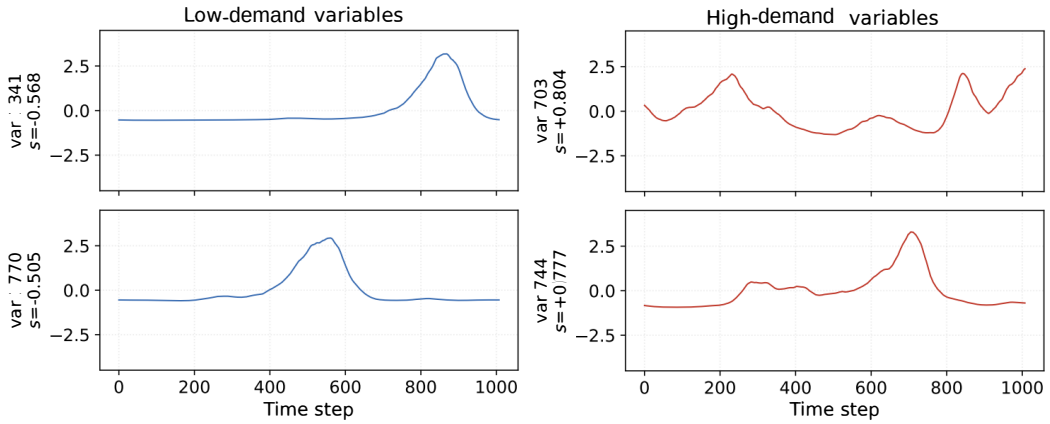


Figure 5: Representative normalized variables with low and high demand scores estimated by the shared bottleneck demand analyzer.