LEARNING LESS IS MORE: ADAPTIVE MODULAR FOUNDATION MODEL FOR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Many existing time series forecasting methods assume that input time series exhibit specific properties, such as seasonality, irregularity, or non-stationarity, and design network modules with these assumptions. Within these methods, foundation models often cover network modules with different assumptions, in order to learn input time series with diverse properties. However, such foundation models process each input time series through all modules, regardless of whether their property assumptions match the input, potentially reducing forecasting accuracy. To address this challenge, we propose ReTS, an adaptive modular foundation model for time series forecasting, where only modules with matched property assumptions assemble together for different input time series at run time. First, by decomposing popular time series forecasting models, we reveal the property assumptions of their network modules, and rearrange these models into a unified recurrent activation structure. Based on this unified structure, ReTS is proposed, which adaptively excludes modules with mismatched property assumptions for each input, thereby avoiding imposing properties the input does not exhibit. Furthermore, the activated structures of ReTS can closely resemble the structures of different baselines for different inputs, and even form unprecedented ones. Experiments on eighteen datasets demonstrate competitive performance against nineteen baselines, with 91% less computational cost at inference than the next-smallest pre-trained baseline. Our code is available at https://anonymous.4open.science/r/ReTS-6307/.

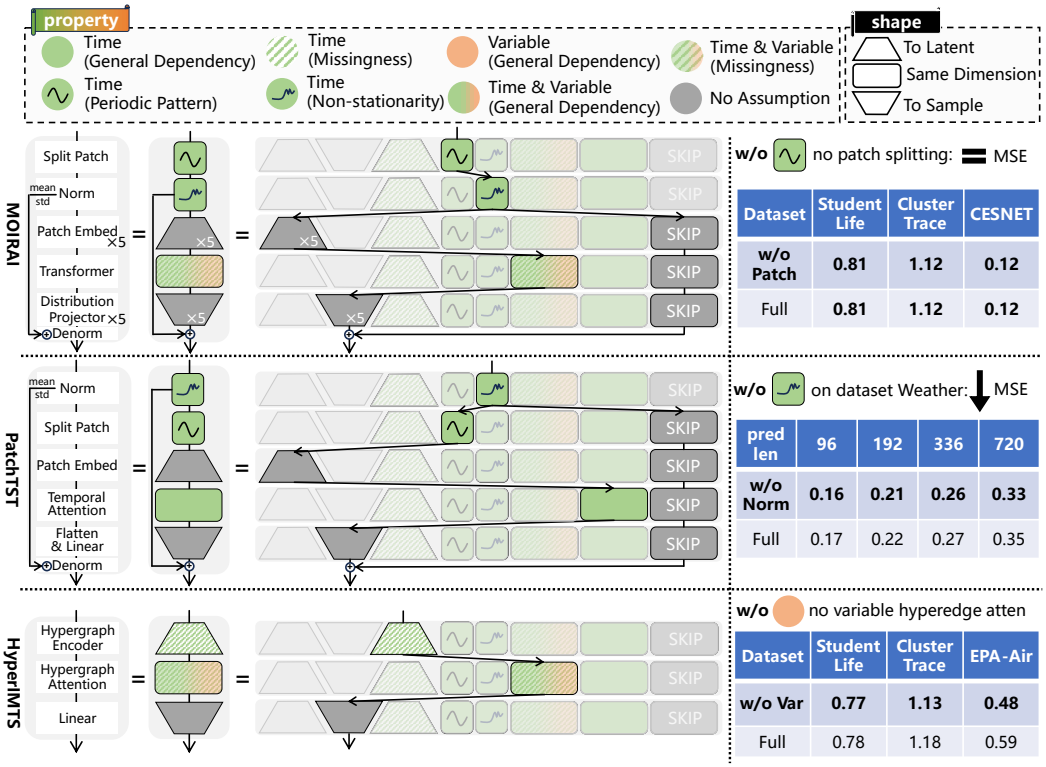
1 INTRODUCTION

Multivariate time series forecasting is essential to domains including healthcare, climate science, energy management, and traffic planning. Inspired by the success of foundation models in natural language processing (NLP), computer vision (CV), and video understanding, time series forecasting foundation models have emerged as a prominent research direction. These models are pre-trained on large-scale time series corpora (Woo et al., 2024; Liu et al., 2025b) or adapt pre-trained weights from other domains such as vision (Chen et al., 2025), enabling zero-shot forecasting across diverse datasets without task-specific retraining. These approaches have advanced universal forecasting capabilities (Shang et al., 2026; Zhang et al., 2025), and most of them scale up the number of parameters to improve the forecasting accuracy, following the scaling laws observed in large language models (LLMs).

However, unlike LLMs, many time series forecasting models embed time series property assumptions in their architectural design. As shown in Figure 1, we illustrate the architecture of three representative models, including pre-trained model MOIRAI (Woo et al., 2024), regular Multivariate Time Series (MTS) model PatchTST (Nie et al., 2023), and Irregular Multivariate Time Series (IMTS) model HyperIMTS (Li et al., 2025). In the left part of the figure, we identify the dominant time series property assumptions of their modules based on well-established standards (Wang et al., 2006; Kottapalli et al., 2025), justified in Appendix B. Specifically, MOIRAI (Woo et al., 2024) mainly assumes time periodicity, missingness, non-stationarity, and variable dependencies in its patching, norm, and masked full-attention Transformer backbones. PatchTST (Nie et al., 2023) mainly assumes time non-stationarity and periodicity in its norm and patching mechanisms. Finally, HyperIMTS (Li et al., 2025) mainly assumes time missingness and variable dependencies in its time series to hypergraph encoder and hypergraph attention modules. More detailed discussions on their justifications are available in Appendix A, and how we decompose and classify more baselines is available in Figure 13.

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.

Based on the decomposition, we can further view the design of these models as selecting modules from a unified large layer recurrently, as illustrated in the middle part of Figure 1 and Figure 2. Thereby, the layer depth in the original model becomes the recurrent depth in the unified structure.



Dataset	Student Life	Cluster Trace	CESNET
w/o Patch	0.81	1.12	0.12
Full	0.81	1.12	0.12

pred len	96	192	336	720
w/o Norm	0.16	0.21	0.26	0.33
Full	0.17	0.22	0.27	0.35

Dataset	Student Life	Cluster Trace	EPA-Air
w/o Var	0.77	1.13	0.48
Full	0.78	1.18	0.59

Figure 1: Decomposition of MOIRAI, PatchTST, and HyperIMTS into their network modules, each annotated with the dominant time series property assumption it encodes. In the middle, models are re-expressed using the unified recurrent structure in Figure 2. On the right, removing modules with mismatched time series assumptions keeps or even lowers the MSE. SKIP indicates residual connections.

With the above decompose-and-select view on the structure of existing models, we found that not all modules are necessary on all types of time series input. As shown in the right part of Figure 1, some modules may even lower the forecasting accuracy on datasets with mismatched time series properties. Specifically, skipping MOIRAI's patching module, which assumes time periodicity, maintains its MSE on three IMTS datasets. Skipping PatchTST's naive normalization, which requires the series mean to be stable, lowers its MSE on the non-stationary Weather dataset. Finally, removing HyperIMTS's variable hyperedge attention module, which mainly assumes variable dependencies, lowers its MSE on the datasets whose variables are mostly linearly uncorrelated. These findings further the discoveries in recent work (Bao et al., 2026), and we provide more details on these phenomena in Appendix A. These assumption mismatches create a fundamental challenge for time series forecasting models: adding more modules to cover diverse properties also increases interference from mismatched ones.

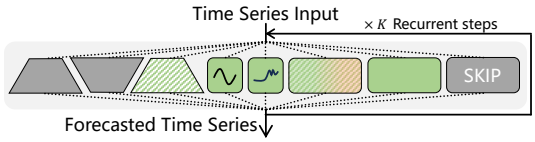


Figure 2: Illustration of the unified recurrent activation structure. By activating different modules recurrently, it can express the structure of many existing time series forecasting models, as illustrated in the middle part of Figure 1 and Figure 13.

To address the challenge of mismatched property assumptions, we propose ReTS, an adaptive modular foundation model for time series forecasting. ReTS rearranges network modules into a unified recurrent activation structure, with each module associated with a specific property assumption

about the input time series. It learns the property-aware representation of each input, and every module owns a learnable module activator that activates the module only when its property assumption holds for the input. Through property-guided recurrent module activation, ReTS organizes the activated modules and adjusts the recurrence depth to provide valid forecasting results. This design ensures that only modules with matching property assumptions process each input, eliminating interference from mismatched modules while reducing the computational cost. The time series properties covered by ReTS are based on well-established standards (Wang et al., 2006; Kottapalli et al., 2025), where we exclude the ones that do not fit the foundation model for time series forecasting. We justify our selections in detail in Appendix B.

Our main contributions are as follows:

- We identify the dominant time series property assumptions encoded by common forecasting modules, where removing modules with mismatched assumptions may keep or even improve forecasting accuracy.
- We propose ReTS, an adaptive modular foundation model for time series forecasting, which adaptively excludes modules with mismatched property assumptions at run time.
- Experiments on eighteen datasets demonstrate competitive performance against nineteen baselines, where ReTS shows outstanding efficiency.

2 Related Work

2.1 Pre-training for Zero-shot Forecasting

Instead of training separate models for each dataset, pre-training for zero-shot forecasting has emerged as a promising paradigm by learning universal representations from a large-scale pre-training dataset (Kottapalli et al., 2025; Liu et al., 2025b; Fei et al., 2026). MOIRAI (Woo et al., 2024) is pre-trained on LOTSA with over 27 billion observations. Sundial (Liu et al., 2025c) predicts the next patch distribution with a flow-matching objective, so it operates on continuous values without discrete tokenization. TimesFM (Das et al., 2024) pre-trains a decoder-style model with input patching on a large corpus of real and synthetic time series. Zeus (Fu et al., 2026) is a tuning-free foundation model built on a multi-scale Transformer with point-wise tokenization. Despite these advances, existing pre-training models may embed fixed property assumptions about the input time series into network architectures, which can cause inputs to be processed by mismatched modules that lower forecasting accuracy. The discussion between channel-dependent and channel-independent approaches also derives from such a prior time series property assumption on variable dependencies (Cheng et al., 2026; Nie et al., 2023).

2.2 Pre-train + Fine-tune for Time Series Forecasting

By adapting pre-trained representations to target datasets, domain shift can be partially mitigated. PrimeNet (Chowdhury et al., 2023) pre-trains on IMTS using time-sensitive contrastive learning, then fine-tunes on downstream forecasting or classification tasks. Chronos (Ansari et al., 2024) tokenizes time series values into a fixed vocabulary and trains a transformer-based language model with a cross-entropy loss. VisionTS (Chen et al., 2025) demonstrates that fine-tuning a vision-pre-trained model for just one epoch achieves significant improvement over its zero-shot performance. MSHLLM (Shang et al., 2026) uses multi-scale hypergraph representations to align LLMs with time series data. ISTS-PLM (Zhang et al., 2025) extends PLMs to irregularly sampled time series. Although these models achieve strong performance after fine-tuning, their dense network structure makes the fine-tuning process computationally expensive. Moreover, analyses of such models reveal that a large fraction of their layers is redundant, and that enlarging model size does not consistently improve accuracy (Bao et al., 2026; Qiu et al., 2026).

3 Problem Definition

With a total of T timestamps and V variables, an MTS or IMTS sample can be denoted as a set containing Y observation tuples $\mathbf{S} := \{(t_i, z_i, v_i)|i = 1, ..., Y\}$, where $t_i \in \{0, ..., T\}$, $z_i \in \mathbb{R}$, and

$v_i \in \{1, ..., V\}$ represent the timestamp, observed value, and variable indicator respectively. For the MTS and IMTS forecasting task, the set of forecast queries $\mathbf{Q} := \{q_j | j = 1, ..., Y_Q\}$ containing Y_Q observations is derived by combining (t_j, v_j) of the j-th observation tuple within the forecast horizon. We aim to learn a forecasting model $\mathcal{F}(\cdot)$, such that given an input MTS or IMTS sample $\mathbf{S}$ and the forecast queries $\mathbf{Q}$, it accurately predicts the corresponding observed values $\mathbf{Z}$:

$$\mathcal{F}(\mathbf{S}, \mathbf{Q}) \rightarrow \mathbf{Z}. \tag{1}$$

4 METHODOLOGY

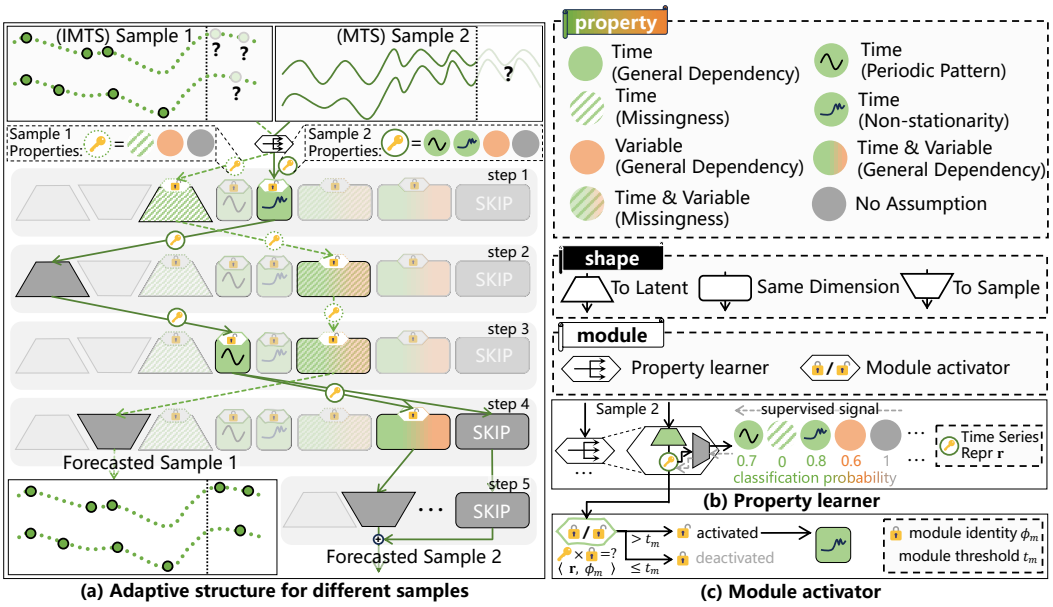


Figure 3: The architecture of ReTS. (a) Modules are dynamically activated for different inputs, with time series representation as "key" and each module's activator as "lock". (b) The property learner maps each input to a property-aware representation $\mathbf{r}$, supervised by time series property labels. (c) The module runs when its property assumption holds for the input and is skipped otherwise.

The overview of ReTS is illustrated in Figure 3. We first describe how ReTS learns a property-aware representation of the input time series in Section 4.1. Subsequently, we describe the property-guided recurrent module activation that uses this representation to activate the modules of the backbone in Section 4.2. The process for obtaining forecast values and training ReTS is described in Section 4.3. Finally, we delineate ReTS from existing methods in Section 4.4.

4.1 PROPERTY-AWARE TIME SERIES REPRESENTATION

In this section, we explain how ReTS maps the input time series into its representation $\mathbf{r}$. Overall, as illustrated in Figure 3 (b), the "key" $\mathbf{r}$ is obtained by the property learner. During pre-training, $\mathbf{r}$ is supervised by the time series property classification labels via a classification head. Therefore, $\mathbf{r}$ can be viewed as the latent space representation of property classification labels. Specifically, given an input time series sample $\mathbf{S}$ over the L-step lookback window, we denote by $\mathbf{X} \in \mathbb{R}^{L \times V}$ the observed values and by $\mathbf{M} \in \{0, 1\}^{L \times V}$ the corresponding binary observation mask, indicating actual observations with 1s and zero-padded or missing values with 0s. The input time series $\mathbf{X}$ is mapped to its latent representation $\mathbf{h} \in \mathbb{R}^{V \times d}$ via the encoder of the property learner:

$$\mathbf{h} = \text{Pool}\left(\text{GELU}\left(\text{Conv}(\mathbf{X} \odot \mathbf{M})\right)\right). \tag{2}$$

The property learner then maps the per-variable latents to one d-dimensional block per property, each produced by a property-specific multilayer perceptron, and concatenates these N_p blocks along

the hidden dimension to form the property-aware representation $\mathbf{r} \in \mathbb{R}^{N_p d}$. It should be noted that these operations are applicable to different lookback window lengths L and numbers of variables V, enabling ReTS to handle input time series of any shape from heterogeneous domains.

After obtaining representation $\mathbf{r}$, we use the time series property classification task for its supervised learning. Specifically, ReTS adopts linear classification heads that map $\mathbf{r}$ into $\hat{\mathbf{p}} \in [0,1]^{N_p}$, estimating the score of the N_p different time series properties. The detailed computation of the predicted score $\hat{\mathbf{p}}$ and the ground truth score $\mathbf{p}$ is described in Appendix C.3. After obtaining the ground truth $\mathbf{p}$, the property learner is trained with a binary cross-entropy loss:

$$\mathcal{L}_{\text{prop}} = -\frac{1}{N_p} \sum_{i=1}^{N_p} \left[p_i \log \hat{p}_i + (1 - p_i) \log(1 - \hat{p}_i) \right], \tag{3}$$

where $\hat{p}_i$ and p_i are the predicted and ground truth scores of the i-th property, respectively. Through the loss $\mathcal{L}_{\text{prop}}$, back-propagated gradients help supervise the representation $\mathbf{r}$ to become property-aware. In the following section, we discuss how the trained "key" $\mathbf{r}$ is used for module activation.

4.2 Property-guided Recurrent Module Activation

In this section, we describe how the property-aware representation $\mathbf{r}$ activates the modules in ReTS, and how the recurrence depth is adjusted based on input. Conceptually, the process is very similar to piecing Lego bricks together, and ReTS is even capable of dynamically transforming its structure at run time. As depicted in Figure 2, in each recurrent step, the unified recurrent activation structure can activate the set of modules in parallel. By recurrently invoking this same set of modules, different recurrent steps can activate diverse modules, mimicking the structure of many existing time series forecasting models like the ones in Figure 1 and Figure 13. Based on this unified structure, the core problem for ReTS becomes which modules can be activated, similar to selecting the correct Lego bricks. ReTS solves this by additionally equipping each module with a learnable module activator, holding the module identity $\phi_m \in \mathbb{R}^{N_p d}$ and threshold t_m that act as the "lock" for module m, tuned to the property the module assumes. Whether the property-aware representation $\mathbf{r}$ can "unlock" the module activator is determined by its matching score, as depicted in Figure 3 (c):

$$\text{score}_m = \langle \phi_m, \mathbf{r} \rangle - t_m. \tag{4}$$

The module is activated only when $\text{score}_m > 0$, indicating the input time series' property matches what the module assumes.

After determining which module can activate, the next move is to order the activated modules among different recurrent steps, such that they can generate valid forecasting results. ReTS handles the problem by letting every module declare its own expected order or required input data, such that modules queue up themselves. Conceptually, it is similar to shaping different Lego bricks with diverse connectors, such that bricks can order correctly without losing flexibility. For example, the hypergraph attention module from HyperIMTS (Li et al., 2025) requires the hypergraph as input, so it will naturally queue after the hypergraph encoder which converts time series to hypergraphs. Further details on the modules in ReTS are available in Appendix C.1.

With the order of modules determined, the final move is to determine the number of recurrent steps K in ReTS. Instead of manually defining fixed rules for K, ReTS learns it from the property-aware representation $\mathbf{r}$ within a predefined interval. Specifically, K is chosen from the integer interval $[K_{\min}, K_{\max}]$, where $K_{\min}$ and $K_{\max}$ are the minimum and maximum number of recurrent steps. A dedicated depth head, a single linear layer over the candidate depths, predicts K from the same representation $\mathbf{r}$ that activates the modules:

$$K = K_{\min} + \text{argmax}_{k \in \{0, \ldots, K_{\max} - K_{\min}\}} \left(\mathbf{W}_K \mathbf{r} + \mathbf{b}_K \right)_k, \tag{5}$$

where $\mathbf{W}_K \in \mathbb{R}^{(K_{\max} - K_{\min} + 1) \times N_p d}$ and $\mathbf{b}_K \in \mathbb{R}^{K_{\max} - K_{\min} + 1}$ are the learnable parameters of the head, one score per candidate depth. Since the head reads the same $\mathbf{r}$ that gates the modules, the depth is determined by the same property-aware judgment as the module selection, rather than prescribed in advance. The head is trained to predict the minimum number of steps required to produce forecasting results given an input time series. For example, when ReTS uses the modules from HyperIMTS (Li et al., 2025) to handle an IMTS sample, the hypergraph encoder, hypergraph attention module, and hypergraph decoder need to be activated sequentially, so $K = 3$ is the target in this scenario. Targets of K can be dynamically detected during model inference by checking whether the forecast horizon has predictions in the current recurrent step, so there is no need to manually specify targets.

4.3 TRAINING OF RETS

In this section, we describe how ReTS is pre-trained and fine-tuned. As discussed in the previous section, ReTS recurrently invokes its set of modules K times to get the forecasting result $\hat{\mathbf{Z}}$, used to calculate masked mean squared error:

$$\mathcal{L} = \frac{\sum_{j=1}^{Y_Q} (\mathbf{M}_Q)_j (\hat{z}_j - z_j)^2}{\sum_{j=1}^{Y_Q} (\mathbf{M}_Q)_j}, \tag{6}$$

where $\mathbf{M}_Q \in \{0,1\}^{Y_Q}$ is the binary mask of observed forecast targets, $(\mathbf{M}_Q)_j$ its j-th entry, and z_j the ground truth value of the j-th forecast query out of the Y_Q in total. Since only observed values contribute to the loss, the same objective trains ReTS on MTS and IMTS datasets. During pre-training, $\mathcal{L}$ is combined with $\mathcal{L}_{\text{prop}}$ from Eq. 3 as the pre-training objective. During fine-tuning, both the property learner and the module activators are frozen, so $\mathcal{L}$ is the only fine-tuning objective.

As for the pre-training data, we use all training set samples of the eighteen datasets in Section 5.1.1, covering time series with diverse combinations of properties. Unlike many baselines that require input time series to be univariate or flattened (Liu et al., 2024; 2025b;c), ReTS is capable of handling time series with any number of variables, as discussed in Appendix C.1. Therefore, mixing time series with different numbers of variables into the pre-training data is feasible for ReTS, which eliminates the need for further data pre-processing. Further dataset statistics are available in Appendix H. It should be noted that to prevent data leakage, we adopt the leave-one-out strategy during pre-training when evaluating the zero-shot and few-shot performance of ReTS. That is, if ECL is the dataset to be evaluated, we will drop ECL from the pre-training dataset for fair comparison. The detailed pre-training procedure, including how the module activators are trained in the hard mode and how the leave-one-out strategy produces the evaluation weights, is described in Appendix C.6.

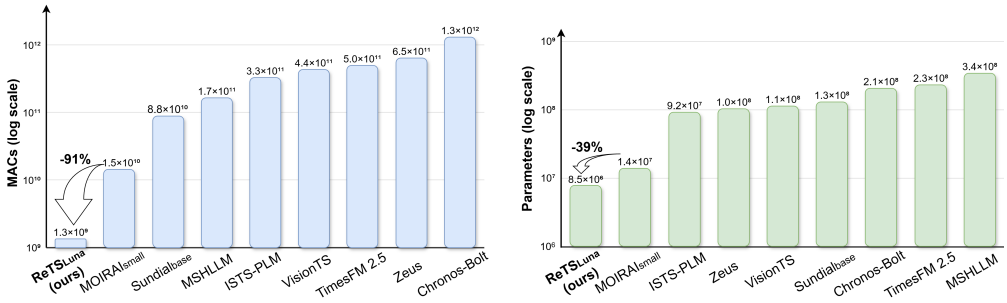
4.4 DELINEATING FROM EXISTING METHODS

We discuss the differences and similarities between ReTS and existing approaches in this section. One category of related methods performs module-level selection or evaluation on task-specific models. TimeRecipe (Zhao et al., 2026) benchmarks the component effectiveness in task-specific models mainly on regular time series datasets. Similarly, CombinationTS (Wang et al., 2026) decomposes MTS forecasting models into orthogonal modules and evaluates each component. ST-LEGO (Li et al., 2026) adopts LLMs to select modules when building traffic prediction models. These models focus on designing task-specific models with fixed structure at run time, and also do not cover irregular time series scenarios. In contrast, ReTS is a more generally applicable foundation model that adaptively changes its optimal model structure given different input time series at run time, which also covers a wider range of time series properties including irregularity or missingness. Benchmark comparisons against TimeRecipe, CombinationTS, and more task-specific baselines are available in Table 8.

Another category performs ensemble learning or model-level selection among multiple pre-trained models. Chroma (Kayaalp et al., 2026) applies ensembling or selection over a set of small pre-trained models to achieve performance competitive with a single large model. SwiftTS (Zhang et al., 2026) identifies the most suitable pre-trained model by computing compatibility scores for unseen datasets. TimeSeriesScientist (Zhao et al., 2025) identifies matched models via agents. These methods operate at the model level, selecting among independently trained whole models. In contrast, ReTS operates at the more fine-grained module level within a single unified architecture, making it possible to adaptively transform into different model structures at run time. Moreover, the property-guided recurrent module activation in ReTS eliminates interference from modules with mismatched property assumptions, a challenge that model-level selection cannot address since each candidate model usually has a fixed structure.

We also discuss the differences against Mixture-of-Experts (MoE) and Neural Architecture Search (NAS). Many MoE models swap experts within a fixed overall structure (Liu et al., 2025a; Shi et al., 2024), where only a few modules are interchangeable. In contrast, ReTS views every network component as interchangeable, as long as they can be pieced together, which enables the transformation of overall structure. NAS searches at the structure level and trains every candidate, which can be a more expensive training process compared to the module-level selection of ReTS. A comparison with NAS is provided in Table 2.

5 EXPERIMENTS



(a) Computational Cost (b) Total Parameters

Figure 4: Comparison of computational cost and model size. Computational cost is averaged over eighteen benchmark datasets. Both panels use a logarithmic scale. $\text{ReTS}_{\text{Luna}}$ has significantly lower computational cost and total parameters relative to the next-smallest model, $\text{MOIRAI}_{\text{small}}$.

5.1 EXPERIMENTAL SETUP

5.1.1 DATASETS

Eighteen time series datasets, covering time series with various combinations of properties from widely studied regular and irregular time series benchmarks, are used in the experiments. For MTS datasets ECL, ETTh1, ETTm1, Traffic, Weather, and ILI, we follow the preprocessing setup of Time-Series-Library (Wang et al., 2024b). For IMTS datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, we adopt the publicly available code pipeline PyOmniTS (Li et al., 2025). For IMTS datasets CESNET, ClusterTrace, EPA-Air, FNSPID, GDELT, RepoHealth, and StudentLife, we follow Time-IMM (Chang et al., 2025). Their introductions and statistics are detailed in Appendix H.

5.1.2 BASELINES

We perform the comparisons also using code pipeline PyOmniTS (Li et al., 2025). Nineteen baselines are included in the benchmark. Main results focus on comparing eight pre-trained methods: MOIRAI (Woo et al., 2024), MSHLLM (Shang et al., 2026), ISTS-PLM (Zhang et al., 2025), Sundial (Liu et al., 2025c), VisionTS (Chen et al., 2025), Chronos (Ansari et al., 2024), TimesFM (Das et al., 2024), Zeus (Fu et al., 2026) Eleven more baselines are available in Appendix E.4. We adapt their publicly available codes into the pipeline for comparisons, where network structures remain unchanged. It should be noted that we have covered more methods in our benchmarking than any of the original papers of the above pre-trained baselines.

5.2 MAIN RESULTS

ReTS is available in three model scales, discussed in Appendix C.5. We use the smallest scale, $\text{ReTS}_{\text{Luna}}$, in all comparisons with baselines. Table 1 compares the zero-shot forecasting results against pre-trained models. Additional benchmarking on few-shot, full-shot, and against task-specific models are available in Appendix E. The visualization of forecasting results can be found in Appendix F. The lookback window and forecast horizon setups of different datasets all follow existing works (Wang et al., 2024b; Li et al., 2025; Chang et al., 2025). As shown in Table 1, ReTS achieves the largest win count in zero-shot evaluation, while using significantly less computational cost. ReTS shows strong performance on IMTS datasets, while closely catching up with baselines on MTS ones, maintaining its efficiency thanks to its adaptive modular structure at run time. It should be noted that all metrics of baselines provided in this paper are based on our replication experiments, using the unified pipeline of PyOmniTS (Li et al., 2025) for fair comparisons and follow corresponding models' original hyperparameter settings for their best accuracy. Therefore, by comparing the metrics reported in baselines' original papers, the zero-shot Table 1, and the full-shot Table 7, we find that zero-shot metrics reported in some baselines' own papers tend to be closer to

Dataset	**MOIRAI** (2024)		**MSHLLM** (2026)		**ISTS-PLM** (2025)		**Sundial** (2025)		**VisionTS** (2025)		**Chronos** (2024)		**TimesFM** (2024)		**Zeus** (2026)		**ReTS$_{\text{Luna}}$** (Ours)		
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	ΔMACs
ECL	0.67	0.58	0.96	0.79	1.02	0.83	-	-	1.37	0.91	-	-	-	-	0.86	0.77	**0.56**	**0.53**	**-98.2%**
ETTh1	0.70	0.53	0.78	0.60	1.11	0.80	**0.47**	**0.43**	1.03	0.69	0.52	0.44	-	-	0.71	0.57	0.66	0.54	**-99.4%**
ETTm1	0.83	0.56	0.79	0.58	1.12	0.80	**0.54**	**0.46**	0.99	0.65	1.03	0.59	-	-	0.72	0.56	0.65	0.51	**-99.4%**
Traffic	-	-	1.60	0.86	1.49	0.80	-	-	2.34	1.07	**0.75**	**0.33**	-	-	-	-	1.13	0.62	**-100.0%**
Weather	0.34	0.35	0.30	0.33	0.64	0.61	**0.29**	**0.30**	0.35	0.35	0.41	0.33	-	-	**0.29**	0.32	**0.29**	**0.30**	**-99.4%**
ILI	-	-	5.17	1.68	7.08	1.90	-	-	6.34	1.83	9.04	1.89	**2.78**	**1.05**	-	-	2.86	1.10	**-99.5%**
MIMIC-III	0.68	0.49	0.89	0.63	0.99	0.67	0.63	0.47	0.87	0.64	0.66	0.49	0.60	0.44	0.90	0.64	**0.42**	**0.37**	**-98.9%**
MIMIC-IV	0.34	0.38	0.61	0.57	0.68	0.62	0.38	0.37	0.61	0.55	0.40	0.38	0.32	0.32	0.65	0.59	**0.22**	**0.28**	**-99.4%**
PhysioNet'12	0.50	0.49	0.51	0.51	0.81	0.67	0.35	0.39	0.55	0.51	0.35	0.38	0.35	0.37	0.42	0.44	**0.28**	**0.35**	**-97.1%**
Human Activity	0.08	0.17	3.24	1.42	4.25	1.70	0.06	0.15	2.48	1.22	**0.05**	0.13	0.06	**0.12**	2.48	1.30	**0.05**	**0.12**	**-94.9%**
USHCN	0.58	0.51	0.63	0.57	0.60	0.53	-	-	0.67	0.60	-	-	0.39	0.35	-	-	**0.19**	**0.25**	**-96.6%**
CESNET	1.02	**0.78**	1.05	0.81	1.00	**0.78**	1.07	0.80	1.41	0.95	1.14	0.80	1.24	0.83	1.01	0.79	**0.98**	**0.78**	**-92.8%**
ClusterTrace	1.12	0.77	1.07	0.76	1.25	0.82	1.01	0.74	1.23	0.81	1.20	0.79	0.94	0.69	-	-	**0.89**	**0.68**	**-90.9%**
EPA-Air	0.47	0.54	0.49	0.54	1.20	0.94	**0.21**	0.32	0.46	0.54	0.22	**0.31**	0.33	0.35	0.34	0.46	0.33	0.45	**-67.8%**
FNSPID	0.14	0.23	0.28	0.42	1.90	1.21	**0.12**	0.20	0.42	0.52	**0.12**	0.20	**0.12**	**0.19**	0.13	0.22	**0.12**	0.20	**-83.8%**
GDELT	1.05	0.69	1.06	0.70	**1.03**	0.69	1.07	0.69	1.19	0.77	1.12	0.66	1.10	**0.65**	1.05	0.70	**1.03**	0.69	**-99.1%**
RepoHealth	0.62	0.42	0.75	0.57	1.26	0.91	**0.53**	0.35	1.07	0.74	0.55	0.35	**0.53**	**0.33**	0.63	0.42	0.56	0.39	**-98.4%**
StudentLife	0.82	0.65	0.84	0.67	0.86	0.70	0.85	0.66	0.94	0.72	0.88	0.66	0.91	0.65	0.83	0.66	**0.76**	**0.64**	**-88.6%**
# Wins	0	1	0	0	1	1	6	3	0	0	3	2	3	5	1	0	**12**	**10**	

Table 1: Zero-shot forecasting results evaluated using MSE and MAE across 18 datasets. Lower values are better. Datasets used during pre-training are excluded from evaluation for the corresponding models and are denoted by a dash (-), justified in Appendix E.3. **Bold** values indicate the best results, and underlined values indicate the second-best results. The ΔMACs column reports the percentage reduction in computational cost of ReTS relative to the best-performing baseline on each dataset. Metrics for ECL, ETTh1, ETTm1, Traffic, and Weather are averaged over forecasting horizons $\{96, 192, 336, 720\}$. Metrics for ILI are averaged over horizons $\{12, 24, 36, 48\}$.

models' full-shot metrics instead of the actual zero-shot ones using their open-sourced weights. In other words, we observe that open-source weights sometimes do not reproduce the zero-shot numbers reported in the original papers. We suspect that the problem may be related to their open-sourced weights, where these pre-trained baselines often have multiple variants of different sizes, while only a few variants have their weights open-sourced. We have already double-checked that their weights are fully downloaded and correctly loaded.

5.3 Efficiency Analysis

We compare the computational cost and total parameters of ReTS$_{\text{Luna}}$ against eight pre-trained baselines in Figure 4, where computational cost is reported in multiply-accumulate operations and model size in total parameters, both measured on the frozen models over the eighteen benchmark datasets. ReTS$_{\text{Luna}}$ uses 91% less computational cost, and 39% fewer parameters, compared to the next-smallest model MOIRAI$_{\text{small}}$. This reduction stems from the adaptive modular structure of ReTS, executing only the modules matching input properties and skipping the computation of mismatched modules, whereas the compared foundation models run their full architecture on every input. Additional efficiency comparisons based on training time and GPU memory usage on a per-dataset basis are provided in Appendix E.5.

5.4 Adaptive Structure of ReTS at Run Time

ReTS adaptively changes its model structure at run time based on the properties of the input time series, as illustrated in Figure 5. For some datasets, the activated structure closely resembles the best-performing task-specific baselines in Table 8, achieving comparable accuracy. For other datasets with complex combinations of properties, like FNSPID, ReTS forms structures that no baseline uses. Unlike the fixed-structure models discussed in Section 4.4, all of these structural changes occur at run time without any retraining.

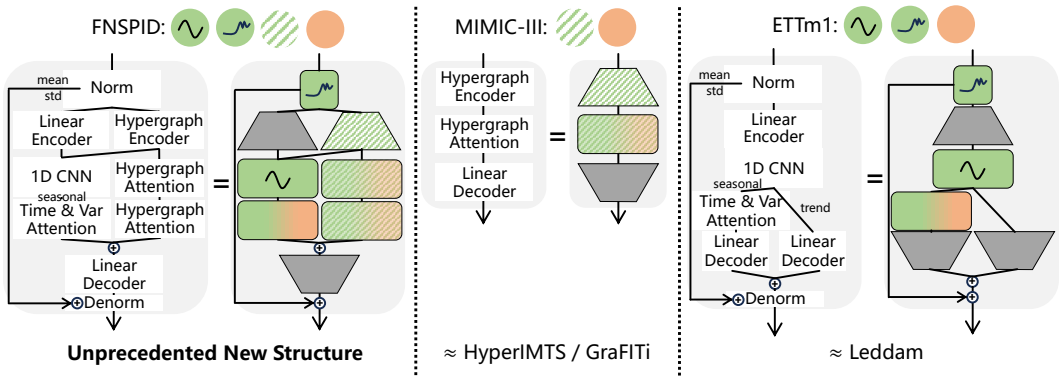


Figure 5: ReTS$_{\text{Luna}}$ adaptively changes its structure based on input properties at run time, and the top-three variants are depicted here. Some closely resemble task-specific baselines, while others form unprecedented module combinations. Other activated structures are covered in Appendix C.4.

Average Metrics	**w/o activator**		**fixed-depth**		**w/o periodicity**		**w/o missingness**		**w/o non-stationarity**		**w/o var**		**w/ NAS**		**ReTS$_{\text{Luna}}$**	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
(18 datasets)	1.46	0.69	0.73	0.52	0.81	0.56	0.88	0.57	1.30	0.72	0.99	0.63	0.69	0.50	**0.67**	**0.49**

Table 2: Average zero-shot ablation results across the eighteen datasets, following setup in Table 1. Detailed results available in Table 9

5.5 Ablation Study

We evaluate the contribution of each design component of ReTS$_{\text{Luna}}$ through seven ablation variants, all tested on the same set of eighteen datasets under identical experimental settings. (1) **w/o activator** removes the learned module activator and runs the full mode on every input, i.e., all eligible modules are activated without per-input module selection; (2) **fixed-depth** replaces the learned recurrence depth with a hand-authored rule that fixes the depth from the detected properties; (3) **w/o periodicity** removes all modules that assume time periodicity; (4) **w/o missingness** removes all modules that assume time missingness; (5) **w/o non-stationarity** removes all modules that assume time non-stationarity; (6) **w/o var** removes all modules that assume variable dependencies; (7) **w/ NAS** replaces the property learner and module activator with neural architecture search. The ablation results are summarized in Table 9, with the results averaged across the eighteen datasets reported in Table 2. As shown in the table, each of the six variants that disables or removes a component degrades the performance across the majority of datasets, confirming that the module activator, the learned recurrence depth, and each of the four properties contribute to the forecasting accuracy of ReTS. The results from **w/o activator** demonstrate the value of per-input module selection. The results from **fixed-depth** show that the learned depth is critical. The results from **w/o periodicity**, **w/o missingness**, **w/o non-stationarity**, and **w/o var** show the necessity of considering the four time series properties. The results from **w/ NAS** demonstrate that the property-aware designs in ReTS used for module activations achieve similar performance against NAS, without searching through numerous candidate structures.

6 Conclusion

This paper introduces ReTS, an adaptive modular foundation model for time series forecasting. We decompose modules from existing forecasting models into a unified recurrent activation structure, revealing that forecasting accuracy can be kept or even improved by removing modules with mismatched property assumptions. Based on this finding, ReTS learns a property-aware representation of each input, and only activates modules with matched properties. Extensive experiments demonstrate that ReTS achieves competitive performance with outstanding efficiency. Also, we provide evidence that ReTS can adaptively transform its structure at run time. Regarding the limitation, ReTS focuses on determining which category of modules to use given the input properties, rather than which module within a category performs best. We will investigate this question in future work.

AI USE STATEMENT

In this work, we used generative AI tools to polish writing and assist coding. We have not used generative AI tools for any other purpose. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the Language of Time Series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Anthony Bao, Venkata Hasith Vattikuti, Jeffrey B. Lai, and William Gilpin. Universal Redundancies in Time Series Foundation Models. In *International Conference on Machine Learning*, 2026.

Ching Chang, Jeehyun Hwang, Yidan Shi, Haixin Wang, Wei Wang, Wen-Chih Peng, and Tien-Fu Chen. Time-IMM: A Dataset and Benchmark for Irregular Multimodal Multivariate Time Series. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Mouxiang Chen, Lefei Shen, Zhuo Li, Xiaoyun Joy Wang, Jianling Sun, and Chenghao Liu. VisionTS: Visual Masked Autoencoders Are Free-Lunch Zero-Shot Time Series Forecasters. In *International Conference on Machine Learning*, 2025.

Hanyin Cheng, Xingjian Wu, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. CoRA: Boosting Time Series Foundation Models for Multivariate Forecasting through Correlation-aware Adapter. In *International Conference on Learning Representations*, 2026.

Ranak Roy Chowdhury, Jiacheng Li, Xiyuan Zhang, Dezhi Hong, Rajesh K. Gupta, and Jingbo Shang. PrimeNet: Pre-training for Irregular Multivariate Time Series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7184–7192, 2023. ISSN 2374-3468.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*, pp. 10148–10167, 2024.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 459–469, 2023. ISBN 979-8-4007-0103-0.

Robert F. Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50(4):987–1007, 1982. ISSN 0012-9682.

Jingru Fei, Kun Yi, Alex Xing Wang, Qingsong Wen, Xiangxiang Zhu, and Wei Fan. Olivia: Harmonizing Time Series Foundation Models with Power Spectral Density. In *International Conference on Machine Learning*, 2026.

Yisong Fu, Zezhi Shao, Chengqing Yu, Yujie Li, Yongjun Xu, Xueqi Cheng, and Fei Wang. Zeus: Towards Tuning-Free Foundation Model for Time Series Analysis. In *International Conference on Machine Learning*, 2026.

Ben D. Fulcher and Nick S. Jones. Highly Comparative Feature-Based Time-Series Classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, 2014. ISSN 1041-4347.

James D. Hamilton. *Time Series Analysis*. 1994.

Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):160035, 2016. ISSN 2052-4463.

Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1, 2023. ISSN 2052-4463.

Mert Kayaalp, Ali Caner Turkmen, Oleksandr Shchur, Pedro Mercado, Abdul Fatir Ansari, Michael Bohlke-Schneider, and Bernie Wang. Test-Time Efficient Pretrained Model Portfolios for Time Series Forecasting. In *International Conference on Learning Representations*, 2026.

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.

Siva Rama Krishna Kottapalli, Karthik Hubli, Sandeep Chandrashekhara, Garima Jain, Sunayana Hubli, Gayathri Botla, and Ramesh Doddaiah. Foundation Models for Time Series: A Survey. *arXiv preprint*, 2025.

Boyuan Li, Yicheng Luo, Zhen Liu, Junhao Zheng, Jianming Lv, and Qianli Ma. HyperIMTS: Hypergraph Neural Network for Irregular Multivariate Time Series Forecasting. In *International Conference on Machine Learning*, 2025.

Shuhao Li, Weidong Yang, Yue Cui, Lipeng Ma, Yixuan Li, Chaoteng Wu, Lu Qin, and Fan Zhang. ST-LEGO: Large Language Models as Modular Architects for Traffic Prediction. In *Proceedings of the ACM Web Conference 2026*, pp. 5040–5051, 2026. ISBN 979-8-4007-2307-0.

Shengsheng Lin, Weiwei Lin, Xinyi Hu, Wentai Wu, Ruichao Mo, and Haocheng Zhong. CycleNet: Enhancing Time Series Forecasting through Modeling Periodic Patterns. *Advances in Neural Information Processing Systems*, 37:106315–106345, 2024.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Junnan Li, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-MoE: Empowering Time Series Foundation Models with Sparse Mixture of Experts. In *International Conference on Machine Learning*, 2025a.

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*, 2022.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative Pre-trained Transformers Are Large Time Series Models. In *International Conference on Machine Learning*, pp. 32369–32399, 2024.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-XL: Long-Context Transformers for Unified Time Series Forecasting. In *International Conference on Learning Representations*, 2025b.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A Family of Highly Capable Time Series Foundation Models. In *International Conference on Machine Learning*, 2025c.

Ziqi Liu. FreqMoE: Enhancing Time Series Forecasting through Frequency Decomposition Mixture of Experts. In *International Conference on Artificial Intelligence and Statistics*, 2025.

Yicheng Luo, Bowen Zhang, Zhen Liu, and Qianli Ma. Hi-Patch: Hierarchical Patch GNN for Irregular Multivariate Time Series. In *International Conference on Machine Learning*, 2025.

S. Makridakis, S. Wheelwright, R. Hyndman, and Y. Chang. *Forecasting: Methods and Applications*. New York, 1998.

Ivan Marisca, Cesare Alippi, and Filippo Maria Bianchi. Graph-based Forecasting with Missing Data through Spatiotemporal Downsampling. In *International Conference on Machine Learning*, pp. 34846–34865, 2024.

M.J. Menne, C.N. Williams, Jr., and R.S. Vose. Long-term daily and monthly climate records from stations across the contiguous united states (u.s. historical climatology network). 1 2016.

Giangiacomo Mercatali, Andre Freitas, and Jie Chen. Graph Neural Flows for Unveiling Systemic Interactions Among Irregularly Sampled Time Series. *Advances in Neural Information Processing Systems*, 37:57183–57206, 2024.

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.

Xin Qiu, Junlong Tong, Yirong Sun, Yunpu Ma, and Xiaoyu Shen. Revealing Scaling Paradox in Large-scale Time Series Models: Implications for More Efficient and Accurate Forecasting. In *International Conference on Machine Learning*, 2026.

Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Zongjiang Shang, Dongliang Cui, Binqing Wu, and Ling Chen. Multi-Scale Hypergraph Meets LLMs: Aligning Large Language Models for Time Series Analysis. In *International Conference on Learning Representations*, 2026.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-Scale Time Series Foundation Models with Mixture of Experts. In *International Conference on Learning Representations*, 2024.

Satya Narayan Shukla and Benjamin Marlin. Multi-Time Attention Networks for Irregularly Sampled Time Series. In *International Conference on Learning Representations*, 2021.

Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting In-Hospital Mortality of ICU Patients: The PhysioNet/Computing in Cardiology Challenge 2012. *Computing in cardiology*, 39:245–248, 2012.

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*, 2024a.

Xiaorui Wang, Fanda Fan, Chenxi Wang, Yuxuan Yang, Rui Tang, Kuoyu Gao, Simiao Pang, Yuanfeng Shang, Zhipeng Liu, Wanling Gao, Lei Wang, and Jianfeng Zhan. CombinationTS: A Modular Framework for Understanding Time-Series Forecasting Models. In *International Conference on Machine Learning*, 2026.

Xiaozhe Wang, Kate Smith, and Rob Hyndman. Characteristic-Based Clustering for Time Series Data. *Data Mining and Knowledge Discovery*, 13(3):335–364, 2006. ISSN 1573-756X.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. Deep Time Series Models: A Comprehensive Survey and Benchmark. *arXiv preprint*, 2024b.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified Training of Universal Time Series Forecasting Transformers. In *International Conference on Machine Learning*, pp. 53140–53164, 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*, 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.

Vijaya Krishna Yalavarthi, Kiran Madhusudhanan, Randolf Scholz, Nourhan Ahmed, Johannes Burchert, Shayan Jawed, Stefan Born, and Lars Schmidt-Thieme. GraFITi: Graphs for Forecasting Irregularly Sampled Time Series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(15):16255–16263, 2024. ISSN 2374-3468.

Guoqi Yu, Jing Zou, Xiaowei Hu, Angelica I. Aviles-Rivero, Jing Qin, and Shujun Wang. Revitalizing Multivariate Time Series Forecasting: Learnable Decomposition with Inter-Series Dependencies and Intra-Series Variations Modeling. In *International Conference on Machine Learning*, 2024.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, 2023. ISSN 2374-3468.

Jiawen Zhang, Shun Zheng, Wei Cao, Jiang Bian, and Jia Li. Warpformer: A Multi-scale Modeling Approach for Irregular Clinical Time Series. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3273–3285, 2023.

Tengxue Zhang, Biao Ouyang, Yang Shu, Xinyang Chen, Chenjuan Guo, and Bin Yang. SwiftTS: A Swift Selection Framework for Time Series Pre-trained Models via Multi-task Meta-Learning. In *International Conference on Learning Representations*, 2026.

Weijia Zhang, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. Irregular Multivariate Time Series Forecasting: A Transformable Patching Graph Neural Networks Approach. In *International Conference on Machine Learning*, 2024.

Weijia Zhang, Chenlong Yin, Hao Liu, and Hui Xiong. Unleashing The Power of Pre-Trained Language Models for Irregularly Sampled Time Series. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 3831–3842, 2025. ISBN 979-8-4007-1454-2.

Yunhao Zhang and Junchi Yan. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. In *International Conference on Learning Representations*, 2023.

Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. TimeSeriesScientist: A General-Purpose AI Agent for Time Series Analysis. *arXiv preprint*, 2025.

Zhiyuan Zhao, Juntong Ni, Shangqing Xu, Haoxin Liu, Wei Jin, and B. Aditya Prakash. TimeRecipe: A Time-Series Forecasting Recipe via Benchmarking Module Level Effectiveness. In *International Conference on Learning Representations*, 2026.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, 2021. ISSN 2374-3468.

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*, pp. 27268–27286, 2022.

A Additional Motivation Justification

We provide additional justifications on our motivation here, on why removing operations or modules with mismatched property assumptions can keep or even improve forecasting accuracy in Figure 1. Recent work has already proven that many modules in existing time series foundation models are redundant (Bao et al., 2026), and our findings further reveal their relationships with time series properties, and demonstrate the results on more task-specific time series forecasting models. It should be noted that our hypothesis only focuses on the cases where the input does not have a certain property. It is possible that even when the input has the property, removing the corresponding module still improves performance, but these are not the cases we focus on. It should also be noted that given the numerous number of time series forecasting models and datasets, it is nearly impossible to guarantee that the conclusion always holds. However, what we can provide in this justification are the ablation results upon several well-known baselines and benchmarking datasets.

A.1 General Temporal Dependency Assumption

This is an assumption that naturally holds for time series forecasting. The feasibility of forecasting relies on the assumption that future progressions can be inferred from past observations (Hamilton, 1994), which is exactly what the general temporal dependency describes. Therefore, there is no "without temporal dependency" case that can be used to justify our motivation.

A.2 Temporal Periodicity Assumption Mismatch

We hypothesize that, on time series datasets with no obvious time periodicity patterns, forecasting accuracy can keep the same without using modules that assume time periodicity. We test this hypothesis on the datasets ClusterTrace, StudentLife, and CESNET, where 23.4%, 19.8%, and 0% of samples carry seasonality, as reported in Table 11. Therefore, CESNET is the one with no obvious time periodicity, and we expect models to keep the same forecasting accuracy, while the other two datasets with different percentages are included here for further comparisons. The periodicity detector is the same as the one ReTS uses, explained in Section 4.1. Overall, we find that modules assuming time periodicity, including patch splitting in MOIRAI (Woo et al., 2024) and PatchTST (Nie et al., 2023), the one dimensional convolution in Leddam (Yu et al., 2024), the moving average in DLinear (Zeng et al., 2023), and the warping function in Warpformer (Zhang et al., 2023), all support our hypothesis. The reason why these modules are classified as time periodicity is provided in Appendix B.

MOIRAI In Figure 1, we skip the input patching module of MOIRAI on the IMTS datasets ClusterTrace, StudentLife, and CESNET, and observe that its MSE is maintained. By removing patching, we mean the lookback window of the input time series is never split into smaller patches, but rather the whole series is directly fed into MOIRAI. The pre-trained weights of MOIRAI can be fully loaded, since MOIRAI can internally zero-pad the input series to its desired length and ignore paddings during masked attention. The patching module groups the observations of each lookback window into fixed-length patches before feeding them to the backbone.

PatchTST We similarly remove the patch splitting operations of PatchTST, so that the whole lookback window is directly embedded without being split into patches. On ClusterTrace, StudentLife, and CESNET, the MSE values change by 1.3%, 0.0%, and -2.5%, respectively, remaining mostly the same. These MSE fluctuations also match the percentage of samples that have a clear periodic pattern, where ClusterTrace matches the minor performance degradation when patching is removed, while CESNET matches the minor performance improvement.

Leddam We further prove the hypothesis for another type of module that can also be used to handle time periodicity, the one-dimensional convolution. We remove the convolution that is used for seasonal-trend decomposition in Leddam, and feed both the trend and seasonal part learning with the non-decomposed series. On ClusterTrace, StudentLife, and CESNET, the MSE values change by 0.9%, 0.0%, and -0.1%, respectively, remaining mostly the same. These MSE fluctuations also match the percentage of samples that have a clear periodic pattern, where ClusterTrace matches the minor performance degradation when the convolution is removed, while CESNET matches the minor

performance improvement. This confirms that the seasonal-trend decomposition of Leddam is also dispensable on time series with no obvious time periodicity, since the convolution has no periodic structure to decompose on such inputs.

DLinear We further test the hypothesis on DLinear, whose moving average decomposes each lookback window into a seasonal part and a trend part, a decomposition that presumes a repeating seasonal pattern. We remove the moving average and feed the non-decomposed series to both the trend and the seasonal linear branches. On ClusterTrace, StudentLife, and CESNET, the MSE values change by 0.9%, 1.0%, and 0.4%, respectively, remaining mostly the same. The degradation is minor and comparable across the three datasets, which is consistent with their weak time periodicity, and confirms that the moving-average decomposition is dispensable on time series with no obvious time periodicity.

Warpformer We finally test the hypothesis on Warpformer, whose warping function remaps the sampling rate of the series into multiple scales, where the finer scales are expected to expose periodic patterns. We remove the warping function and feed the original series directly to the backbone. On ClusterTrace, StudentLife, and CESNET, the MSE values change by 4.6%, -0.5%, and -0.2%, respectively, remaining mostly the same. These MSE fluctuations match the percentage of samples that have a clear periodic pattern, where 23.4% for ClusterTrace, the dataset with the most seasonal samples, matches the largest performance degradation when the warping function is removed, while 0% for CESNET matches the near-neutral change. This confirms that the warping function of Warpformer, which assumes periodic patterns at the finer scales, is also dispensable on time series with no obvious time periodicity, like CESNET.

A.3 Temporal Non-stationarity Assumption Mismatch

We hypothesize that, on time series datasets where mean or standard deviation (std) of observed values changes significantly over time, forecasting accuracy may improve without using normalization. We test this hypothesis on the Weather dataset over all four forecasting horizons, where the means of the lookback window and forecast horizon largely differ. Therefore, we expect models to improve forecasting accuracy on the Weather dataset without using normalization. Overall, we find that models using normalization operations that assume a stable mean, including PatchTST with its naive normalization and Leddam with RevIN (Kim et al., 2022), all support our hypothesis. The reason why the normalization is classified as time non-stationarity is provided in Appendix B.

PatchTST In Figure 1, we remove the relatively naive normalization operations in PatchTST on the regular time series dataset Weather, and observe a consistent performance improvement over multiple forecasting horizons. The normalization module of PatchTST rescales each lookback window to a zero mean and unit variance and re-anchors the forecast to the input mean. This operation assumes that the level and the scale of the series are stable across the boundary from the lookback window to the forecast horizon, so that the input mean and variance are good predictors of the horizon mean and variance. We test this assumption by measuring, on the raw data, the fraction of the channels that are non-stationary under the Augmented Dickey-Fuller unit-root test, and deem the input non-stationary when this fraction reaches 0.5. The non-stationarity detector is the same as the one ReTS uses, explained in Section 4.1, and the per-sample detection rates it produces are reported in Table 11. On the Weather dataset, the per-channel fraction reaches 0.85, and removing the module changes the MSE by -0.9% to -5.3% across the forecast horizons. On ILI, the per-channel fraction reaches 0.92, and removing the module also improves the accuracy.

Leddam Leddam adopts the more advanced normalization method RevIN (Kim et al., 2022), which is exactly motivated by the limitation of naive normalization on non-stationary inputs. However, ablation shows that Leddam's performance can still benefit from removing the operations upon the mean of the observed values inside RevIN. On the Weather dataset, removing these mean operations changes the MSE by -0.9% to -4.5% across the four forecasting horizons. This confirms that even RevIN, which is designed to handle non-stationary inputs, still relies on a mean stability assumption when re-anchoring the forecast to the input mean, and removing that operation improves the accuracy on the non-stationary Weather dataset.

A.4 Temporal Missingness Assumption

Unfortunately, removing the missingness or irregularity assumption in most existing IMTS models will not lead to any difference on fully observed datasets. Their modules mostly rely on the binary mask to indicate missingness or padding values, with 1s indicating observed values and 0s otherwise. For fully observed time series, the mask will just be filled with 1s. It means there is no need to ignore any missing data, and the behavior regresses to exactly that of models that do not consider missingness. As for bipartite graphs (Yalavarthi et al., 2024) or hypergraphs (Li et al., 2025) that directly represent time series as sparse graphs, they just regress to fully connected ones. Therefore, the time missingness assumption in most existing IMTS models essentially adds extra logic to handle the special case of missingness, without affecting their structure on fully observed time series. All we need to do is ensure modules with the missingness assumption are prioritized over the others when handling IMTS input.

A.5 Variable Dependency Assumption Mismatch

We hypothesize that variable dependency learning modules can be removed when most variables carry uncorrelated information. It should be noted that since variable correlations may be exposed in high-order spaces, which is difficult to statistically discover, our ablation here focuses on testing linearly uncorrelated cases. The set of linearly uncorrelated cases subsumes the fully unrelated ones, and we want to demonstrate the possible forecasting accuracy improvement after removing these modules. We test the case where the variables are mostly linearly uncorrelated on the datasets EPA-Air, ClusterTrace, and StudentLife, where only 8.9%, 19.0%, and 37.1% of total samples are detected as having linear correlations among variables, as reported in Table 11. The variable dependency detector is the same as the one ReTS uses, explained in Section 4.1. We expect that removing these variable dependency learning modules may keep or even improve the forecasting accuracy on these datasets. Overall, we find that the variable-to-variable hyperedge message passing in HyperIMTS (Li et al., 2025), the cross-dimension stage in Crossformer (Zhang & Yan, 2023), the variable dimension MLP in TSMixer (Ekambaram et al., 2023), and the attention along variable axis in Leddam (Yu et al., 2024), all support our hypothesis. The reason why these modules are classified as assuming variable dependencies is provided in Appendix B.

HyperIMTS In Figure 1, we remove the variable-to-variable message passing among hyperedges in HyperIMTS and observe that on the datasets EPA-Air, ClusterTrace, and StudentLife, the MSE is kept or even lowered. The variable-to-variable message passing module of HyperIMTS lets the variable hyperedges exchange information across variables, an operation that is useful mainly when the variables carry related information. As contrasting examples, on the datasets whose variables are highly linearly correlated, including MIMIC-III and PhysioNet'12, removing the module raises the MSE by 6.6% and 6.1%.

Leddam Leddam applies attention along the variable axis, which assumes variable dependencies. On the mostly linearly uncorrelated datasets EPA-Air, ClusterTrace, and StudentLife, removing the module changes the MSE by -2.2%, -1.3%, and -0.6%, respectively. These improvements also align with our hypothesis.

Crossformer We further test the hypothesis on Crossformer, whose two-stage attention layer contains a cross-dimension stage that models variable dependencies. We remove the cross-dimension stage and keep only the cross-time stage. On the mostly linearly uncorrelated datasets EPA-Air, ClusterTrace, and StudentLife, removing the module changes the MSE by -8.0%, -1.2%, and -0.0%, respectively. The improved accuracy is also consistent with our hypothesis, and the biggest improvement on EPA-Air even aligns with the fact that it has the least percentage of samples that have linear correlations among variables.

TSMixer We finally test the hypothesis on TSMixer, which mixes the variables with a multilayer perceptron applied along the variable axis. We remove this variable dependency learning and keep only the mixing along the time axis. On the mostly linearly uncorrelated datasets EPA-Air, ClusterTrace, and StudentLife, removing the module changes the MSE by -6.2%, -1.0%, and -0.4%, respectively. Again, the improved results align with our hypothesis.

B Property Detection Justification

We justify the four time series properties detected by ReTS: time periodicity, time non-stationarity, time missingness, and variable dependencies. It should be noted that our property detection is not meant to cover all possibilities, which is nearly impossible, but rather is based on existing works and excludes the ones that are not applicable to a foundation model in multivariate time series forecasting. During code implementation, we also have modules for general dependency, matching any time series input that does not expose these properties. Our selected properties are based on the highly cited work of Wang (Wang et al., 2006), which applied nine feature measurements on time series, including trend, seasonality, serial correlation, non-linearity, skewness, kurtosis, self-similarity, chaos, and periodicity. Specifically, trend indicates a long-term change in the mean level (Makridakis et al., 1998), falling under the definition of time non-stationarity (Liu et al., 2022). Seasonality, defined as a pattern that repeats itself over fixed intervals of time (Makridakis et al., 1998), falls under time periodicity. Serial correlation falls under variable dependencies, an extensively discussed property under multivariate time series forecasting task (Kottapalli et al., 2025). Nonlinearity is learnable in most modern neural networks (Woo et al., 2024; Nie et al., 2023; Li et al., 2025), thus not explicitly detected in ReTS. Skewness and kurtosis require assessing data distribution (Wang et al., 2006), which is not feasible for zero-shot inference of a foundation model. Self-similarity reveals long-range dependence (Wang et al., 2006), falling under time periodicity. Chaos can be hard to identify without prior knowledge on the underlying dynamic system and multiple trajectories (Wang et al., 2006), and is therefore also not feasible in zero-shot inference situations.

We also discuss the broader time series properties evaluated in other highly cited papers (Fulcher & Jones, 2014; Kottapalli et al., 2025). Our selected time series properties align well with those discussed by Kottapalli (Kottapalli et al., 2025) except for high dimensionality, which describes the data volume of the time series dataset rather than a property of the individual sample. Higher-moment properties, such as conditional heteroscedasticity and heavy tails (Engle, 1982), require analyzing the data distribution and are therefore not feasible in zero-shot forecasting. As for the signal-to-noise ratio, it cannot be estimated without the true signal.

All eighteen benchmark datasets, used in regular and irregular time series forecasting evaluations (Nie et al., 2023; Li et al., 2025; Chang et al., 2025), can be characterized by the covered properties.

We finally justify, operation by operation, the classification of the module operations considered in existing forecasting models. An operation is assigned to a property when its computation presumes that property of the input, so that running the operation on a mismatched input adds interference rather than information. It should be noted that some operations upon timestamp or mask inputs, like positional encoding, are not covered here, since our motivation mainly discusses the properties upon the observed values of time series inputs.

Moving average The moving average separates a repeating seasonal component from the mean level and is therefore classified as time periodicity, as in the seasonal decomposition of Autoformer (Wu et al., 2021).

Patch/segment Patching groups consecutive timestamps into local tokens. Usually, each is assumed to carry a recurring local pattern, and it is therefore classified as time periodicity, as in PatchTST (Nie et al., 2023). It should be noted that patching is a widely adopted technique that may be used for other purposes, like reducing computational cost (Zhang et al., 2024). This work mainly discusses its impact on time series without clear time periodicity, not its other aspects.

(Masked) attention Attention weighs positions along one axis of the input. Masked attention uses a mask to indicate missingness or padding values, so it can handle time missingness (Woo et al., 2024). When used without a mask and applied along the time axis, it models temporal dependencies without presuming a specific property, and belongs to the general class. When applied along the variable axis, it models inter-variable dependencies and is therefore classified as variable dependencies (Zhang & Yan, 2023).

Learnable parameter Learnable parameters are used to capture a recurrent cycle in CycleNet (Lin et al., 2024). Therefore, they can embed the time periodicity assumption.

Table 3: The property assumptions encoded by the modules in existing popular time series forecasting models, and representative methods for each module. The reason for these classifications is explained in Appendix B. It should be noted that some modules can express more properties, and what is listed here only indicates their most widely applied property assumptions.

Property Assumption	Module	Representative Method
Time (General Dependency)	Attention (w/o mask)	Crossformer (Zhang & Yan, 2023)
	Linear/MLP	CycleNet (Lin et al., 2024), DLinear (Zeng et al., 2023), TSMixer (Ekambaram et al., 2023)
	Recurrence	HD-TTS (Marisca et al., 2024)
Time (Periodic Pattern)	Moving Average	Autoformer (Wu et al., 2021), DLinear (Zeng et al., 2023), TimeMixer (Wang et al., 2024a)
	Patch/Segment	MOIRAI (Woo et al., 2024), PatchTST (Nie et al., 2023), Hi-Patch (Luo et al., 2025)
	Learnable Parameter	CycleNet (Lin et al., 2024)
	FFT	FEDformer (Zhou et al., 2022)
	Convolution	Leddam (Yu et al., 2024), TimesNet (Wu et al., 2023)
	Warping Function	Warpformer (Zhang et al., 2023)
Time (Non-stationarity)	Normalization	RevIN (Kim et al., 2022)
Time (Missingness)	Attention (with mask)	MOIRAI (Woo et al., 2024)
	Graph	HyperIMTS (Li et al., 2025), GraFITi (Yalavarthi et al., 2024)
	Gating	tPatchGNN (Zhang et al., 2024)
	XDE	Latent-ODE (Rubanova et al., 2019)
	Timestamp Delta	Warpformer (Zhang et al., 2023)
Variable (General Dependency)	Attention	Crossformer (Zhang & Yan, 2023), MOIRAI (Woo et al., 2024), Timer-XL (Liu et al., 2025b)
	Linear/MLP	TSMixer (Ekambaram et al., 2023)
	Graph	Hi-Patch (Luo et al., 2025), HyperIMTS (Li et al., 2025), GraFITi (Yalavarthi et al., 2024)
	Convolution	TimesNet (Wu et al., 2023)

Linear/MLP When used for dimension transformations in the encoder or decoder (Yu et al., 2024), they are generally not assigned a property learning task, and we classify these cases as no assumption. As for the rest cases, the dependency depends on the axis of application. When applied along the time axis, they learn a mixing of past observations (Lin et al., 2024). When applied along the variable axis (Ekambaram et al., 2023), they capture variable dependencies.

FFT The fast Fourier transform is widely used in time series analysis to identify seasonality (Zhou et al., 2022), and is therefore classified as time periodicity.

Graph Graph message passing propagates information over related nodes. For the hypergraph in HyperIMTS (Li et al., 2025), it directly expresses IMTS data without padding, so it handles irregular sampling and is classified as time missingness, while its hyperedges encode either temporal or inter-variable dependencies.

Recurrence Recurrent propagation accumulates information across timestamps, models temporal dependencies without presuming a specific property, and belongs to the general temporal dependency class.

Convolution A one-dimensional convolution along the time axis smooths out the series, and can be used in seasonal-trend decomposition (Yu et al., 2024), and is therefore classified as embedding time periodicity. When applied along the variable axis in 2D, it can mix variables and is classified as variable dependencies (Wu et al., 2023).

Gating Gating ignores missingness and allows observed data, so it can be used to encode IMTS (Zhang et al., 2024).

XDE An XDE integrates the latent state at the exact observation times and can handle non-uniform sampling by construction, and is therefore classified as time missingness, as in Latent-ODE (Rubanova et al., 2019) and GRU-ODE-Bayes (De Brouwer et al., 2019).

Warping function A warping function can remap the sampling rate of a series to coarser ones (Zhang et al., 2023), where the finer scales can show periodic patterns, and the coarser scales show overall trends.

Normalization Naive normalization rescales each lookback window to a reference level and scale (Nie et al., 2023), which presumes that the mean and variance of the series are stable across the lookback window and the forecast horizon. RevIN (Kim et al., 2022) tackles the time non-stationarity by introducing learnable terms, but can still benefit from our discovery mentioned in Appendix A.3.

Timestamp delta The delta-time encoding represents the gaps between consecutive observations explicitly and can be used to handle time missingness, as in Warpformer (Zhang et al., 2023).

C Additional Details on ReTS

Our code implementation is available at https://anonymous.4open.science/r/ReTS-6307/, in case any of the descriptions below are not specific enough.

C.1 Modules Used in ReTS

We explain what modules are used in ReTS, and why these modules are selected. It should be noted that ReTS focuses on answering "what category of modules to use given the properties of input time series" instead of "which module is the best performing one given the properties of input time series". That is, ReTS is primarily designed to distinguish the property assumption column in Table 3, instead of directly choosing the best in the module column.

As shown in Figure 6, ReTS comes with three different variants, namely $\text{ReTS}_{\text{Luna}}$, $\text{ReTS}_{\text{Terra}}$, and ReTS_{Sol}. All variants are equipped with modules that can cover the property assumptions in Table 3:

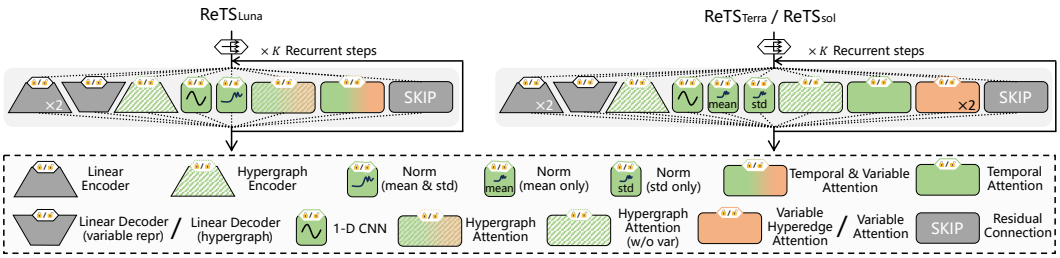


Figure 6: Architecture of the three ReTS variants, ReTS$_{\text{Luna}}$, ReTS$_{\text{Terra}}$, and ReTS$_{\text{Sol}}$. ReTS$_{\text{Terra}}$ has more fine-grained module division and property detection than ReTS$_{\text{Luna}}$: the normalization module is divided into mean and std, and the variable hyperedge attention is separated from the hypergraph attention module. ReTS$_{\text{Sol}}$ has larger hidden dimension of 512 compared to 384 in ReTS$_{\text{Terra}}$.

(a) Time (General Dependency): the temporal attention, whose key–value pairs are built along the time axis and whose attention is not masked by missingness, models general temporal dependencies, as in many transformer-based baselines (Zhang & Yan, 2023; Wu et al., 2021); (b) Time (Periodic Pattern): the one-dimensional convolution along the time axis separates trend from seasonality, as in Leddam (Yu et al., 2024); (c) Time (Non-stationarity): the normalization module RevIN (Nie et al., 2023), which we improve to handle time missingness as well as any number of variables in input time series; (d) Time (Missingness): by converting input IMTS into hypergraphs, it not only handles time missingness, but is also capable of processing any number of variables in input (Li et al., 2025); (e) Variable (General Dependency): the variable hyperedge attention (Li et al., 2025) and the variable attention (Zhang & Yan, 2023; Wu et al., 2021), whose key–value pairs are built along the variable axis, capture variable dependencies.

C.1.1 Module Interface

We define the module interface on six aspects: the input tensors it requires, the tensors it updates, its position constraints, its dependency requirements, its module identity, and its threshold. These function like the connector shape of Lego bricks, determining how different bricks are pieced together. We follow each module's common usage patterns in existing baselines to design these interfaces (Li et al., 2025; Yu et al., 2024; Yalavarthi et al., 2024; Zhang & Yan, 2023; Wu et al., 2021). Table 4 summarizes the interface of the eight modules in ReTS$_{\text{Luna}}$, and the other variants follow the same interface pattern.

Module	Type	Requires	Updates
hypergraph encoder	encoder	$\mathbf{X}$, $\mathbf{M}$	hypergraph repr.
linear encoder	encoder	$\mathbf{X}$	variable repr.
RevIN	learner	$\mathbf{X}$, $\mathbf{M}$	normalized input, prediction
1-D CNN	learner	variable repr.	trend, seasonal repr.
temporal & variable attention	learner	seasonal repr.	seasonal repr.
hypergraph attention	learner	hypergraph repr.	hypergraph repr.
linear decoder (variable)	decoder	trend, seasonal repr.	prediction
linear decoder (hypergraph)	decoder	hypergraph repr.	prediction

Table 4: Module interface of the eight modules in ReTS$_{\text{Luna}}$, and the residual connection module "SKIP" is not listed here. "Requires" lists the tensors a module needs in order to run, and "Updates" lists the tensors it provides to the modules in the next recurrence.

On general position constraints, encoders can only be activated before decoders. On dependency requirements, a module can only be activated if the module types it depends on have already been activated in earlier recurrent steps. Specifically, the hypergraph attention depends on the hypergraph encoder, which guarantees that the input is encoded and refined before it is decoded. The hypergraph encoder further declares the assumption of time missingness, and is purged before activation whenever the input has no missing values. The RevIN module is stateful and must appear in both the first and the last recurrent steps, to normalize the input and denormalize the prediction.

C.2 Module Activator

We provide more details on how the module activator in $\text{ReTS}_{\text{Luna}}$ changes the activated structure at run time. As in Section 4.2, each module m is activated when its activation score $\text{score}_m = \langle \phi_m, \mathbf{r} \rangle - t_m$ in Eq. 4 is positive, so the activated set depends on the property-aware representation $\mathbf{r}$ of the input. Across the eighteen benchmark datasets, this property-guided activation produces two dominant structures. For fully observed MTS inputs, the activated structure of ReTS is similar to Leddam (Yu et al., 2024). For most IMTS inputs, the activated structure of ReTS is similar to HyperIMTS (Li et al., 2025) or GraFITi (Yalavarthi et al., 2024). A few IMTS datasets have all four properties, time periodicity, time non-stationarity, time missingness, and variable dependencies, and the activated structure is a combination not used by the included baselines, visualized in Figure 5. The rest of the activation variants mostly differ by diverse numbers of hypergraph attention layers and the presence of a RevIN layer or a 1-D CNN layer. In extremely rare cases, the activated modules in ReTS may not produce forecasting results, and ReTS will automatically fall back to running the full structure at maximum depth. We statistically check the percentage of this fallback, and find that only 1.2% of samples in the Weather dataset trigger the fallback among all eighteen datasets.

C.3 Property Learner Details

We provide more details on the two components of the property learner: how the representation $\mathbf{r}$ is mapped to the predicted property score $\hat{\mathbf{p}} \in [0,1]^{N_p}$, and how the ground truth property score $\mathbf{p} \in [0,1]^{N_p}$ is calculated.

As discussed in Section 4.1, the representation $\mathbf{r}$ contains one d-dimensional block for each of the N_p properties. Let $\mathbf{r}_i \in \mathbb{R}^d$ be the block of $\mathbf{r}$ for property i. The property learner maps each block to a score in $[0,1]$ with a property-specific linear head and a sigmoid function:

$$\hat{p}_i = \sigma\left(\mathbf{w}_i^\top \mathbf{r}_i + b_i\right), \tag{7}$$

where $\mathbf{w}_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$ are the learnable parameters of the i-th head and $\sigma(\cdot)$ is the sigmoid function. Stacking the N_p heads gives the predicted score $\hat{\mathbf{p}} \in [0,1]^{N_p}$, which is trained by the binary cross-entropy loss in Eq. 3. Since the head is linear over the property-specific block, each property is read from its own part of $\mathbf{r}$ and the properties do not interfere with each other.

The ground truth score of each property is calculated from the input time series and its observation mask $\mathbf{M} \in \{0,1\}^{L \times V}$, following the detectors introduced in Section 4.1. Time periodicity is measured by how much of the spectral energy of each variable concentrates on its dominant frequency. Let $f_{v,k}$ be the spectral power of variable v at frequency k, and k_v be the frequency with the largest power; the score is

$$p_{\text{periodic}} = \max_{1 \le v \le V} \frac{f_{v,k_v}}{\sum_k f_{v,k}}. \tag{8}$$

Time non-stationarity is measured by the fraction of variables whose unit root cannot be rejected. Let $\hat{\tau}_v$ be the ADF statistic of variable v and c_α the critical value at the significance level α, at which the unit root cannot be rejected when $\hat{\tau}_v > c_\alpha$; the score is the fraction of variables with a finite statistic whose unit root cannot be rejected,

$$p_{\text{nonstat}} = \frac{1}{V_{\text{test}}} \sum_{v \in \mathcal{V}_{\text{test}}} \mathbb{1}\left[\hat{\tau}_v > c_\alpha\right], \tag{9}$$

where $\mathcal{V}_{\text{test}}$ is the set of variables with a finite ADF statistic and $V_{\text{test}} = |\mathcal{V}_{\text{test}}|$. Variable dependency is measured by the redundancy among the variables, derived from the effective dimensionality of their correlation structure. Let $\mu_1, \ldots, \mu_J$ be the positive eigenvalues of the pairwise-complete correlation matrix of the variables, $\lambda_j = \mu_j / \sum_{j'} \mu_{j'}$ the normalized eigenvalue spectrum, and $d_{\text{eff}} = \exp\left(-\sum_j \lambda_j \log \lambda_j\right)$ the effective dimensionality, which is one for perfectly redundant variables and V for independent variables; the score is

$$p_{\text{dep}} = 1 - \frac{d_{\text{eff}}}{V}. \tag{10}$$

Time missingness is read directly from the observation mask, and its score is the fraction of missing values over the full input spanning the lookback window and the forecast queries:

$$p_{\text{miss}} = 1 - \frac{1}{LV + Y_Q} \sum_{t} \sum_{v} M_{t,v}, \tag{11}$$

where $M_{t,v}$ ranges over the LV lookback observations and the Y_Q forecast queries. Each score is bounded in $[0, 1]$ and grows with the strength of the corresponding property, matching the range of the predicted score $\hat{\mathbf{p}}$.

As for the classification accuracy of the property learner, it achieves an average of 90.9% AUC for time periodicity, 82.5% AUC for time non-stationarity, and 83.7% AUC for variable dependencies over all the datasets. Since the targets are heavily class-imbalanced, F1 at the 0.5 sigmoid threshold can be a misleading metric and is therefore not reported here.

C.4 Adaptive Structure of ReTS at Run Time

We additionally describe the structures that $\text{ReTS}_{\text{Luna}}$ can activate at run time. We denote the three structures in Figure 5 as (a), (b), and (c), from left to right. The remaining activated structures are small variations of (a)-(c): (d) is (b) with "Norm" added to the first recurrence, observed on the CESNET dataset; (e) is (c) without "Norm", observed on the Weather dataset. Overall, structures (a)-(c) are the most commonly observed. Structure (a) lets ReTS become one of the best models on the FNSPID dataset in the full-shot evaluations in Table 7 and Table 8. (b) and (c) closely resemble the existing baselines HyperIMTS (Li et al., 2025) and Leddam (Yu et al., 2024), which also reflects why HyperIMTS is very competitive on MIMIC-III and Leddam wins on multiple MTS datasets.

C.5 ReTS Variants Comparisons

Table 5 compares the three scales of ReTS on the same eighteen datasets under both zero-shot and full-shot settings, and three observations stand out. First, all three scales remain highly competitive across the datasets, and $\text{ReTS}_{\text{Luna}}$, the smallest variant, is not clearly worse than the larger ones. This is consistent with the design of ReTS, where property-guided recurrent module activation already removes the modules whose property assumptions do not match the input. Second, the full-shot metrics on EPA-Air show the most obvious difference among the three variants. Since $\text{ReTS}_{\text{Terra}}$ and ReTS_{Sol} divide their variable dependency learning modules more finely than Luna does, datasets with weak variable dependencies, like EPA-Air as reported in Table 11, can benefit from removing these modules. It also matches our motivation. Third, the zero-shot metrics of the largest variant, ReTS_{Sol}, fall behind the other two variants on most IMTS datasets. The possible reason is that, under the exact same learning rate, early stopping patience, and pre-training dataset size, ReTS_{Sol} is less trained than the other two smaller variants, but catches up after full-shot training.

C.6 Pre-training Procedures

We describe the full pre-training procedure of ReTS, and how it enables the training of hard-gated module activator. As mentioned in Section 4.3, the pre-training data is the training portion of the eighteen benchmark datasets, and the pre-training objective is $\mathcal{L}$ plus the property score loss $\mathcal{L}_{\text{prop}}$ from Eq. 3. In practice, the pre-training is divided into three stages that freeze different parts of the model in turn. In stage one, the backbone runs in the full mode at the maximum recurrence depth $K_{\max}$, so that all eligible modules are active on every input and every module body receives the forecasting loss. The property learner is additionally supervised by $\mathcal{L}_{\text{prop}}$ against the ground truth property scores, so that the representation $\mathbf{r}$ becomes property-aware before the module activators are learned. In stage two, the property learner is frozen, and the backbone runs in the soft mode at a per-input target depth. In soft mode, every eligible module runs with its contribution scaled by the sigmoid of its score, so the forecasting loss trains the module activators in addition to the module bodies. The depth head is additionally supervised by a cross-entropy term that is added to the forecasting loss, to reproduce the target depth. In stage three, the property learner and the module activators are frozen, the depth head is applied directly to predict the depth, and the backbone runs in the hard mode, so that the module bodies are fine-tuned through the sparse computation graph that the gates and the depth select for each input. To summarize, we first train the module bodies on a working full architecture, then train the gates and the depth head by soft weighting, and finally adapt

Dataset	**ReTS$_{\text{Luna}}$**				**ReTS$_{\text{Terra}}$**				**ReTS$_{\text{Sol}}$**			
	zero-shot		full-shot		zero-shot		full-shot		zero-shot		full-shot	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	**0.56**	**0.53**	**0.17**	**0.25**	0.61	0.54	**0.17**	**0.25**	0.60	0.54	**0.17**	**0.25**
ETTh1	0.66	0.54	0.45	0.44	0.65	0.53	**0.44**	**0.43**	**0.64**	**0.52**	**0.44**	**0.43**
ETTm1	0.65	0.51	**0.39**	0.39	0.63	**0.50**	**0.39**	**0.38**	**0.62**	**0.50**	**0.39**	0.39
Traffic	**1.13**	**0.62**	0.45	**0.29**	1.20	0.65	**0.44**	**0.29**	1.16	0.64	0.45	**0.29**
Weather	0.29	**0.30**	**0.24**	0.27	**0.28**	**0.30**	**0.24**	**0.26**	**0.28**	**0.30**	**0.24**	**0.26**
ILI	2.86	**1.10**	1.65	**0.76**	3.28	1.20	1.66	**0.76**	**2.82**	1.12	**1.62**	**0.76**
MIMIC-III	0.42	**0.37**	0.40	0.37	**0.39**	**0.37**	**0.38**	**0.36**	**0.39**	**0.37**	0.40	0.37
MIMIC-IV	0.22	**0.28**	**0.22**	0.28	0.22	**0.28**	**0.22**	**0.27**	**0.21**	**0.28**	**0.22**	**0.27**
PhysioNet'12	**0.28**	**0.35**	**0.28**	0.35	0.30	0.36	**0.28**	**0.34**	0.30	0.37	**0.28**	**0.34**
Human Activity	**0.05**	**0.12**	**0.04**	**0.11**	**0.05**	**0.12**	**0.04**	**0.11**	**0.05**	0.13	**0.04**	**0.11**
USHCN	0.19	**0.25**	0.18	0.26	**0.18**	**0.25**	**0.17**	0.23	**0.18**	0.27	**0.17**	**0.21**
CESNET	**0.98**	**0.78**	**0.96**	**0.76**	0.99	**0.78**	**0.96**	**0.76**	1.03	**0.78**	**0.96**	**0.76**
ClusterTrace	**0.89**	**0.68**	**0.89**	**0.68**	0.97	0.73	**0.89**	**0.68**	1.04	0.77	**0.89**	**0.68**
EPA-Air	0.33	0.45	0.33	0.45	0.33	**0.44**	0.29	0.39	**0.32**	0.45	**0.24**	**0.36**
FNSPID	0.12	0.20	**0.11**	**0.18**	**0.11**	**0.19**	**0.11**	**0.18**	0.12	0.20	**0.11**	**0.18**
GDELT	**1.03**	0.69	**1.03**	0.69	**1.03**	**0.68**	**1.03**	**0.65**	1.04	0.72	**1.03**	0.69
RepoHealth	**0.56**	**0.39**	0.58	**0.40**	**0.56**	0.40	**0.57**	**0.40**	0.58	0.41	**0.57**	**0.40**
StudentLife	**0.76**	0.64	**0.75**	**0.63**	**0.76**	**0.63**	**0.75**	**0.63**	**0.76**	0.64	**0.75**	**0.63**

Table 5: Zero-shot and full-shot forecasting results of the three ReTS model scales, Luna, Terra, and Sol, ordered from small to large, where Terra has more fine-grained module division and property detection than Luna, and Sol has a larger hidden dimension than Terra. Lower values are better. **Bold** values indicate the best results within each evaluation mode, and underlined values indicate the second-best. Metrics for ECL, ETTh1, ETTm1, Traffic, and Weather are averaged over forecasting horizons $\{96, 192, 336, 720\}$. Metrics for ILI are averaged over horizons $\{12, 24, 36, 48\}$.

the bodies to the hard selection that the learned gates and depth produce. This ordering is the key to training the hard mode of the module activators.

Because of this staged procedure and the modest size of each variant, the pre-training process of ReTS is fast and consumes few resources, runnable even on an NVIDIA GeForce GTX 1080Ti with only 11 GB of GPU memory. Each variant takes approximately twelve hours, and different variants can be trained in parallel. The leave-one-out strategy in Section 4.3 therefore yields eighteen sets of weights for the zero-shot and few-shot evaluation of ReTS, one per benchmark dataset, each pre-trained on the remaining seventeen datasets.

D Implementation Details

We follow the setting of the widely used Time-Series-Library (Wang et al., 2024b) in learning rate adjustments. All experiments run with a maximum of 300 epochs and early stopping patience of 10 epochs. To mitigate randomness, we conduct each experiment with five different random seeds ranging from 2024 to 2028, also in line with the setting of Time-Series-Library, taking the mean of the results. However, since we have conducted comprehensive benchmarking comparisons over eighteen datasets and nineteen baselines, where the table cells can become too small to read if standard deviations are also displayed, we choose to follow the same approach as existing pre-trained baselines' papers (Fu et al., 2026; Liu et al., 2025c; Chen et al., 2025; Ansari et al., 2024), which omit standard deviations in tables. MSE is used as the training loss function for models, unless a custom loss function proposed in the original paper is used. When adapting regular time series models for IMTS, masks indicating observed values are included in the MSE calculations during training. The detailed settings for the hyperparameters are provided in Appendix G. All models are trained on a Linux server with PyTorch version 2.9.1 using one NVIDIA A100 GPU with 80 GB of GPU memory, and efficiency analysis is conducted on another Linux server with the same PyTorch version using one NVIDIA GeForce RTX 5090 GPU with 32 GB of GPU memory.

E Additional Experiments

E.1 Few-Shot Evaluation

Table 6 reports the few-shot forecasting results of the foundation-model baselines against ReTS. In the few-shot setting we randomly sample 10% of each target dataset for fine-tuning and keep all other settings identical to the full-shot evaluation. ReTS achieves the largest number of wins on MSE and is tied for the largest on MAE, even with only a small fraction of each dataset. We also note that some metrics in Table 6 do not follow the expected zero-shot $>$ few-shot $>$ full-shot ordering, in that a few few-shot results are worse than the corresponding zero-shot results. We retried those entries with learning rates one order of magnitude lower and one order of magnitude higher than the original learning rate, but observed no improvement, so these results are not mistakes. For example, $\text{ReTS}_{\text{Luna}}$ performs worse on PhysioNet'12 after few-shot training, but regains its strongest performance after full-shot training. The reason may be related to the characteristic of the medical dataset, where different samples are collected from heterogeneous patients (Silva et al., 2012). Therefore, samples used in few-shot may only cover patients with specific symptoms, causing models to overfit easily.

E.2 Full-Shot Evaluation

Table 7 reports the full-shot forecasting results, in which all models are fine-tuned on the complete training split of each target dataset. $\text{ReTS}_{\text{Luna}}$ even obtains a larger or equal number of wins on both MSE and MAE, compared to the zero-shot and few-shot ones. As mentioned in the few-shot evaluation, some metrics in Table 7 do not follow the expected zero-shot $>$ few-shot $>$ full-shot ordering, in that a few full-shot results are worse than the corresponding few-shot results. We retried those entries with learning rates one order of magnitude lower and one order of magnitude higher than the original learning rate, but observed no improvement, so these results are not mistakes.

Dataset	MOIRAI (2024)		MSHLLM (2026)		ISTS-PLM (2025)		Sundial (2025)		VisionTS (2025)		Chronos (2024)		TimesFM (2024)		Zeus (2026)		$\text{ReTS}_{\text{Luna}}$ (Ours)		
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	ΔMACs
ECL	**0.18**	**0.26**	0.86	0.77	0.26	0.35	-	-	0.20	0.28	-	-	-	-	ME	ME	0.19	0.28	**-98.2%**
ETTh1	0.63	0.52	0.72	0.58	0.75	0.62	**0.45**	0.44	**0.45**	**0.43**	**0.45**	**0.43**	-	-	**0.45**	0.44	0.49	0.47	**-100.0%**
ETTm1	0.42	0.41	0.72	0.55	0.75	0.60	0.40	0.41	0.40	0.41	0.49	0.40	-	-	**0.38**	**0.39**	0.40	0.41	**-100.0%**
Traffic	-	-	1.43	0.81	0.78	0.41	-	-	0.52	**0.31**	ME	ME	-	-	-	-	**0.51**	0.34	**-99.9%**
Weather	0.28	0.28	0.29	0.31	0.29	0.34	0.30	0.31	0.27	0.29	0.27	**0.27**	-	-	0.26	**0.27**	**0.25**	0.28	**-100.0%**
ILI	-	-	4.93	1.64	5.78	1.72	-	-	6.64	1.86	2.06	**0.86**	**1.94**	0.87	-	-	2.09	0.91	**-99.5%**
MIMIC-III	0.68	0.44	0.86	0.62	0.56	0.46	0.63	0.47	0.69	0.48	0.63	0.47	0.57	0.44	0.57	0.44	**0.42**	**0.37**	**-99.7%**
MIMIC-IV	0.29	0.30	0.61	0.57	0.29	0.35	0.35	0.35	0.39	0.40	0.36	0.36	0.32	0.33	0.33	0.33	**0.22**	**0.29**	**-86.9%**
PhysioNet'12	0.38	0.40	0.47	0.50	0.40	0.44	0.35	0.39	0.38	0.44	0.34	0.37	0.33	**0.36**	0.33	0.37	**0.31**	**0.36**	**-97.1%**
Human Activity	0.07	0.16	0.22	0.33	0.20	0.34	0.05	0.12	0.11	0.22	0.05	0.12	**0.04**	**0.11**	0.05	**0.11**	0.05	0.12	**-96.7%**
USHCN	0.23	0.26	0.26	0.34	0.23	0.30	-	-	0.20	0.26	-	-	0.21	0.25	-	-	**0.18**	**0.24**	**-98.4%**
CESNET	1.07	0.78	1.05	0.80	1.00	0.79	1.01	0.78	1.37	0.86	1.04	**0.76**	0.99	0.77	1.01	0.77	**0.97**	**0.76**	**-91.9%**
ClusterTrace	1.21	0.81	1.07	0.76	1.30	0.83	1.01	0.74	1.22	0.82	1.12	0.76	0.96	**0.70**	1.04	0.76	**0.92**	**0.70**	**-90.9%**
EPA-Air	0.34	0.43	0.46	0.53	1.10	0.89	**0.21**	0.32	0.28	0.38	**0.21**	**0.31**	0.26	0.35	0.33	0.45	0.34	0.45	**-91.8%**
FNSPID	0.14	0.23	0.26	0.40	0.86	0.79	0.12	0.20	0.97	0.74	0.12	**0.19**	**0.11**	**0.19**	0.13	0.20	0.12	0.20	**-83.8%**
GDELT	1.10	0.68	1.04	0.69	**1.03**	0.70	1.08	0.71	1.21	0.76	1.09	**0.65**	1.07	0.66	1.05	0.67	1.05	0.68	**-99.1%**
RepoHealth	0.61	0.41	0.73	0.56	1.20	0.89	0.53	0.35	1.01	0.70	0.53	0.35	**0.52**	**0.33**	0.69	0.43	0.56	0.39	**-99.0%**
StudentLife	0.84	0.64	0.81	0.66	0.84	0.69	0.85	0.66	0.86	0.69	0.87	0.65	0.85	0.64	0.83	0.65	**0.75**	**0.63**	**-99.6%**
# Wins	1	1	0	0	1	0	2	0	1	2	2	**7**	4	5	2	3	**9**	**7**	

Table 6: Few-shot forecasting results evaluated using MSE and MAE across 18 datasets. Lower values are better. Datasets used during pre-training are excluded from evaluation for the corresponding models and are denoted by a dash (-), justified in Appendix E.3. 'ME' indicates memory error. **Bold** values indicate the best results, and underlined values indicate the second-best results. The ΔMACs column reports the percentage reduction in computational cost of ReTS relative to the best-performing baseline on each dataset. Metrics for ECL, ETTh1, ETTm1, Traffic, and Weather are averaged over forecasting horizons $\{96, 192, 336, 720\}$. Metrics for ILI are averaged over horizons $\{12, 24, 36, 48\}$.

Dataset	MOIRAI (2024)		MSHLLM (2026)		ISTS-PLM (2025)		Sundial (2025)		VisionTS (2025)		Chronos (2024)		TimesFM (2024)		Zeus (2026)		$\text{ReTS}_{\text{Luna}}$ (Ours)		
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	ΔMACs
ECL	0.18	0.26	0.26	0.35	0.27	0.35	0.21	0.28	0.19	0.28	ME	ME	ME	ME	ME	ME	**0.17**	**0.25**	**-98.2%**
ETTh1	0.59	0.50	0.56	0.51	0.78	0.69	0.46	0.45	**0.43**	**0.43**	0.49	0.46	0.50	0.46	0.46	0.44	0.45	0.44	**-99.9%**
ETTm1	0.49	0.44	0.49	0.46	0.85	0.67	0.42	0.41	**0.38**	0.40	0.45	0.42	0.41	0.41	**0.38**	**0.39**	0.39	**0.39**	**-100.0%**
Traffic	0.53	0.31	0.96	0.52	0.79	0.42	0.48	**0.29**	0.47	**0.29**	ME	ME	ME	ME	ME	ME	**0.45**	**0.29**	**-99.9%**
Weather	0.30	0.28	0.29	0.31	0.29	0.34	0.28	0.30	0.25	0.28	ME	ME	ME	ME	0.26	**0.27**	**0.24**	**0.27**	**-98.3%**
ILI	3.75	0.96	4.83	1.61	6.32	1.74	1.95	0.82	3.64	1.28	1.86	0.80	1.77	0.80	1.98	0.78	**1.65**	**0.76**	**-99.5%**
MIMIC-III	0.66	0.47	0.82	0.62	0.42	0.39	0.62	0.46	0.61	0.43	0.60	0.43	0.56	0.44	0.56	0.43	**0.40**	**0.37**	**-99.7%**
MIMIC-IV	0.28	0.30	0.59	0.57	0.25	0.31	0.34	0.34	0.24	**0.28**	0.36	0.36	ME	ME	0.32	0.32	**0.22**	**0.28**	**-99.8%**
PhysioNet'12	0.34	0.37	0.47	0.50	0.29	0.36	0.34	0.38	0.30	0.36	0.33	0.36	0.32	0.36	0.33	0.37	**0.28**	**0.35**	**-99.1%**
Human Activity	0.07	0.16	0.18	0.32	0.08	0.19	0.05	0.12	**0.04**	0.12	**0.04**	**0.11**	**0.04**	**0.11**	**0.04**	**0.11**	**0.04**	**0.11**	**-96.7%**
USHCN	0.22	0.25	0.27	0.36	0.19	0.28	0.26	0.31	0.21	0.29	0.18	**0.22**	0.26	0.28	**0.17**	**0.22**	0.18	0.26	**-97.8%**
CESNET	1.02	0.77	1.00	0.78	1.00	0.78	0.98	0.76	1.36	0.86	0.97	**0.74**	**0.94**	**0.74**	0.97	0.76	0.96	0.76	**-91.9%**
ClusterTrace	1.12	0.80	1.07	0.76	1.27	0.83	0.92	0.70	1.03	0.76	0.93	0.70	0.93	0.69	1.13	0.81	**0.89**	**0.68**	**-81.3%**
EPA-Air	0.23	0.35	0.45	0.53	0.68	0.65	**0.21**	**0.32**	0.28	0.39	**0.21**	**0.32**	0.23	0.34	0.29	0.41	0.33	0.45	**-67.8%**
FNSPID	0.12	0.21	0.28	0.42	0.17	0.27	0.12	0.19	0.13	0.23	**0.11**	0.19	**0.11**	0.19	0.13	0.20	**0.11**	**0.18**	**-83.8%**
GDELT	1.11	0.67	**1.03**	0.69	**1.03**	0.69	1.07	0.70	1.04	0.69	1.12	**0.65**	1.05	0.69	1.05	0.66	**1.03**	0.69	**-99.1%**
RepoHealth	0.58	0.40	0.72	0.56	0.77	0.63	0.54	0.35	0.83	0.63	0.55	**0.33**	**0.51**	**0.33**	0.66	0.38	0.58	0.40	**-99.0%**
StudentLife	0.81	0.64	0.80	0.65	0.80	0.66	0.85	0.66	0.79	0.66	0.85	0.64	0.83	0.65	0.83	0.65	**0.75**	**0.63**	**-98.6%**
# Wins	0	0	1	0	1	0	1	2	3	3	3	6	4	3	3	4	**12**	**12**	

Table 7: Full-shot forecasting results evaluated using MSE and MAE across 18 datasets. Lower values are better. 'ME' indicates memory error. **Bold** values indicate the best results, and underlined values indicate the second-best results. The ΔMACs column reports the percentage reduction in computational cost of ReTS relative to the best-performing baseline on each dataset. Metrics for ECL, ETTh1, ETTm1, Traffic, and Weather are averaged over forecasting horizons $\{96, 192, 336, 720\}$. Metrics for ILI are averaged over horizons $\{12, 24, 36, 48\}$.

E.3 Zero-Shot and Few-shot Evaluation Justifications

We justify why zero-shot Table 1 and few-shot Table 6 omit some evaluations. It omits a baseline on a dataset whenever that dataset appears in the model's pre-training corpus, following the same evaluation practice as Zeus (Fu et al., 2026). We identified these overlaps by inspecting the pre-training corpora reported by each model's original paper and their code repositories. MOIRAI (Woo et al., 2024) is trained on LOTSA, which includes the CDC FluView and PeMS traffic sources that our ILI and Traffic benchmarks are built from, so MOIRAI is excluded on ILI and Traffic. TimesFM (Das et al., 2024) includes the hourly Electricity, Traffic, and Weather series in its version 1.0. Since we use TimesFM 2.5 for comparison, stated in Appendix G, and version 2.5 additionally includes the GIFT-Eval pre-training dataset, which contains ETT datasets, it is excluded on ECL, ETTh1, ETTm1, Traffic, and Weather. Chronos (Ansari et al., 2024) uses the USHCN dataset in its pre-training datasets and ECL in its in-domain evaluation datasets, so it is excluded on ECL and USHCN. Zeus (Fu et al., 2026) is pre-trained on a corpus that includes the CDC FluView, PeMS traffic, USHCN, and Alibaba ClusterTrace series, so it is excluded on ILI, Traffic, USHCN, and ClusterTrace. Sundial (Liu et al., 2025c) is pre-trained on TimeBench, which combines LOTSA and the Chronos corpus with other sources and therefore inherits the CDC FluView, PeMS traffic, USHCN, and Electricity series. Therefore, we exclude Sundial on ILI, Traffic, USHCN, and ECL. The remaining baselines, MSHLLM, ISTS-PLM, and VisionTS, use text-only or vision-only pre-training that does not include any of our datasets, so all eighteen datasets are retained for them.

E.4 Additional Benchmarking Comparisons

Dataset	**PrimeNet** (2023)		**HD-TTS** (2024)		**Hi-Patch** (2025)		**HyperIMTS** (2025)		**mTAN** (2021)		**FreqMoE** (2025)		**Leddam** (2024)		**PatchTST** (2023)		**GNeuralFlow** (2024)		**TimeRecipe** (2026)		**CombinationTS** (2026)		**ReTS$_{\text{Luna}}$** (Ours)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	1.01	0.83	0.19	0.29	ME	ME	0.26	0.34	7.54	0.86	0.19	0.28	**0.17**	0.27	0.20	0.28	0.42	0.46	0.20	0.28	0.22	0.31	**0.17**	**0.25**
ETTh1	1.11	0.80	0.57	0.54	0.75	0.66	0.73	0.66	1.11	0.80	0.47	0.45	0.45	**0.43**	**0.44**	0.44	1.08	0.80	0.45	0.44	**0.44**	**0.43**	0.45	0.44
ETTm1	1.11	0.79	0.48	0.48	0.92	0.71	0.90	0.71	15.32	1.33	0.83	0.68	**0.39**	0.40	**0.39**	0.40	1.33	0.88	0.40	0.40	0.42	0.41	**0.39**	**0.39**
Traffic	1.48	0.80	0.57	0.30	ME	ME	ME	ME	1.68	0.80	ME	ME	0.46	**0.29**	0.47	0.30	0.84	0.45	0.49	0.31	0.63	0.39	**0.45**	**0.29**
Weather	0.64	0.61	0.26	0.32	0.30	0.34	0.28	0.33	0.63	0.55	ME	ME	0.25	0.28	0.25	0.28	0.46	0.44	0.26	0.28	0.27	0.29	**0.24**	**0.27**
ILI	7.25	1.93	4.97	1.49	6.51	1.77	7.31	1.94	5.82	1.67	1.99	0.90	**1.52**	**0.76**	1.75	0.82	5.77	1.67	2.11	0.92	2.47	1.03	1.65	**0.76**
MIMIC-III	0.90	0.64	0.42	0.39	0.43	0.41	**0.37**	**0.37**	0.85	0.62	0.70	0.55	0.59	0.51	0.57	0.45	0.69	0.54	0.58	0.48	0.68	0.54	0.40	**0.37**
MIMIC-IV	0.63	0.57	0.24	0.29	0.24	0.29	**0.22**	**0.28**	0.45	0.45	0.38	0.41	0.37	0.40	0.29	0.32	0.40	0.40	0.37	0.38	0.43	0.42	**0.22**	**0.28**
PhysioNet'12	0.79	0.68	**0.28**	**0.35**	0.33	0.38	**0.28**	**0.35**	0.37	0.42	0.42	0.46	0.38	0.43	0.34	0.39	0.39	0.44	0.34	0.40	0.39	0.44	**0.28**	**0.35**
Human Activity	1.09	0.43	0.05	0.14	0.05	0.13	**0.04**	**0.11**	0.09	0.21	0.12	0.23	0.09	0.20	0.07	0.17	0.13	0.27	0.11	0.22	0.11	0.23	**0.04**	**0.11**
USHCN	0.46	0.48	**0.16**	0.27	0.23	0.29	0.17	0.27	0.56	0.50	0.38	0.39	0.24	0.31	0.22	0.31	0.26	0.34	0.26	0.32	0.26	0.31	0.18	**0.26**
CESNET	1.01	0.79	0.98	0.77	1.00	0.78	0.99	0.78	1.00	0.79	1.01	0.78	0.97	**0.75**	1.16	0.83	1.03	0.79	0.98	0.77	1.02	0.77	**0.96**	0.76
ClusterTrace	1.27	0.83	1.33	0.85	1.17	0.80	1.18	0.79	1.48	0.91	1.04	0.75	1.21	0.83	1.16	0.80	1.63	0.96	1.09	0.78	1.15	0.80	**0.89**	**0.68**
EPA-Air	1.31	0.99	1.06	0.87	1.11	0.90	0.60	0.62	0.73	0.68	0.25	0.37	0.23	0.35	**0.22**	**0.34**	1.34	0.99	0.24	0.36	0.24	0.35	0.33	0.45
FNSPID	2.27	1.32	0.14	0.24	0.12	0.21	0.13	0.24	0.19	0.30	0.14	0.24	0.13	0.22	0.13	0.23	0.53	0.49	0.13	0.22	0.13	0.23	**0.11**	**0.18**
GDELT	**1.03**	0.69	**1.03**	0.69	**1.03**	0.69	**1.03**	0.69	1.05	**0.68**	1.05	0.70	1.05	0.69	1.10	0.72	**1.03**	**0.68**	1.05	0.70	1.05	0.70	**1.03**	0.69
RepoHealth	1.56	1.01	0.82	0.66	0.71	0.59	0.62	0.52	1.78	1.12	**0.57**	0.43	0.59	0.43	0.60	0.44	2.18	1.24	0.59	0.44	0.59	0.44	0.58	**0.40**
StudentLife	0.88	0.72	0.78	0.65	0.78	0.65	0.78	0.65	0.83	0.68	0.80	0.65	0.80	0.65	0.81	0.65	0.87	0.70	0.80	0.65	0.80	0.65	**0.75**	**0.63**
# Wins	1	0	3	1	1	0	5	4	0	1	1	0	3	4	3	1	1	1	0	0	1	1	**12**	**14**

Table 8: Additional baseline forecasting results evaluated using MSE and MAE across 18 datasets. Lower values are better. 'ME' indicates memory error. **Bold** values indicate the best results, and underlined values indicate the second-best results. Metrics for ECL, ETTh1, ETTm1, Traffic, and Weather are averaged over forecasting horizons $\{96, 192, 336, 720\}$. Metrics for ILI are averaged over horizons $\{12, 24, 36, 48\}$.

We provide additional benchmarking comparisons with eleven more baselines here, where ReTS$_{\text{Luna}}$ adopts its full-shot results. These baselines include: PrimeNet (Chowdhury et al., 2023), HD-TTS (Marisca et al., 2024), Hi-Patch (Luo et al., 2025), HyperIMTS (Li et al., 2025), mTAN (Shukla & Marlin, 2021), FreqMoE (Liu, 2025), Leddam (Yu et al., 2024), PatchTST (Nie et al., 2023), GNeuralFlow (Mercatali et al., 2024), TimeRecipe (Zhao et al., 2026), CombinationTS (Wang et al., 2026). Since the open-sourced weights of PrimeNet are specific to datasets with 41 variables, we retrain the model from scratch on all datasets. Also, since MOIRAI-MoE (Liu et al., 2025a) can hardly be run on the eighteen datasets on our servers due to out-of-GPU-memory errors, we choose the smaller baseline FreqMoE to show ReTS's performance against the MoE-based method. As can be seen, ReTS$_{\text{Luna}}$ gets the highest win count, and is competitive against several strong task-specific baselines. It should be noted that, as discussed in Section 5.2, for the zero-shot Table 1, many existing pre-trained baselines claim to outperform task-specific baselines under zero-shot evaluation, but our replication experiments against their open-sourced weights are not able to reproduce such results. In comparison, their full-shot performance can indeed reach the accuracy claimed in their papers.

Therefore, based on our faithful and unified replication experiments, we honestly conclude that these pre-trained baselines, including ReTS, are still not able to fully beat task-specific ones using zero-shot performance. However, the value of investigating pre-trained methods still holds, where datasets with only a few samples can be quickly generalized from. In contrast, task-specific methods may overfit easily on a small number of samples.

E.5 Additional Efficiency Analysis

We provide additional efficiency comparisons in Figure 7, 8, and 9 using the full-shot model configurations of Table 7, with the training time and GPU memory usage as the resource usage metrics. The forecast horizons are set to 720 for the datasets ECL, ETTh1, ETTm1, Traffic, and Weather, following CycleNet (Lin et al., 2024), and set to 48 for the dataset ILI. The remaining lookback and forecast horizon setups are the same as in Table 7. Since we use separate GPUs for benchmarking models and efficiency analysis, models that cannot be fitted into 32 GB of GPU memory during full-shot training are marked as out of memory during efficiency analysis. However, measurements of computational cost and total parameters in Figure 4 can be performed on frozen models, so we are able to obtain all those metrics.

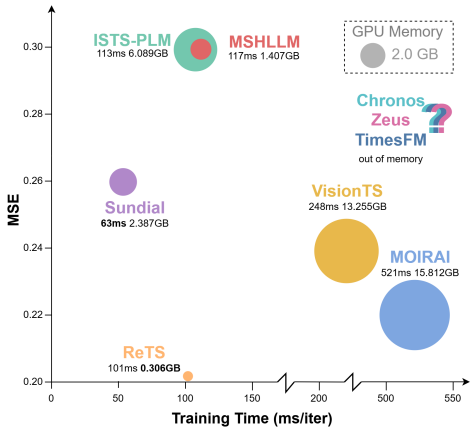
As can be seen from Figure 7, 8, and 9, ReTS uses the least GPU memory on all datasets, except for GDELT. As for the training time, ReTS is among the fastest models, except for a few datasets where it is about the average speed of the baselines. To conclude, combined with the computational cost and total parameter comparisons in Figure 4, ReTS achieves competitive forecasting accuracy with outstanding efficiency against pre-trained baselines.

E.6 Additional Ablation Study

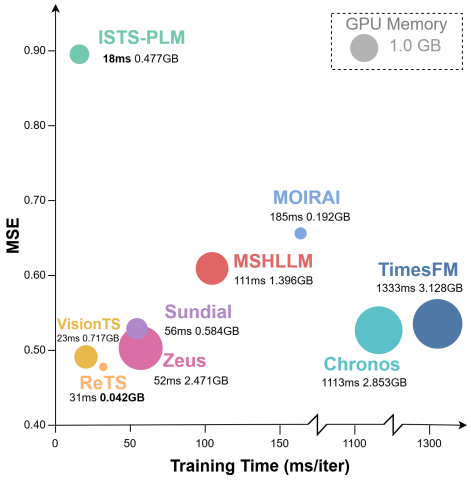
Dataset	**w/o activator**		**fixed-depth**		**w/o periodicity**		**w/o missingness**		**w/o non-stationarity**		**w/o var**		**w/ NAS**		**ReTS$_{\text{Luna}}$**	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	0.68	0.58	0.68	0.58	1.07	0.82	**0.56**	**0.53**	1.26	0.90	1.58	0.98	0.66	0.58	**0.56**	**0.53**
ETTh1	0.74	0.59	0.74	0.59	0.85	0.64	**0.66**	**0.54**	1.51	0.96	1.24	0.79	0.69	0.57	**0.66**	**0.54**
ETTm1	0.74	0.56	0.74	0.56	0.93	0.64	**0.65**	**0.51**	1.72	1.01	1.12	0.71	0.69	0.54	**0.65**	**0.51**
Traffic	1.22	0.70	1.22	0.70	1.67	0.88	**1.13**	**0.62**	1.66	0.86	2.77	1.15	1.20	0.66	**1.13**	**0.62**
Weather	0.30	0.31	0.30	0.31	0.31	0.33	0.29	**0.30**	1.46	0.90	0.45	0.39	**0.28**	**0.30**	0.29	**0.30**
ILI	3.33	1.21	3.33	1.21	3.74	1.37	3.68	1.27	8.52	2.15	3.92	1.30	2.93	1.13	**2.86**	**1.10**
MIMIC-III	4.28	0.97	0.51	0.42	0.44	0.38	0.68	0.51	0.48	0.40	0.46	0.39	0.43	0.38	**0.42**	**0.37**
MIMIC-IV	4.42	0.83	0.29	0.32	**0.22**	**0.28**	0.36	0.37	0.28	0.32	0.23	0.29	**0.22**	**0.28**	**0.22**	**0.28**
PhysioNet'12	0.68	0.54	0.32	0.37	0.31	0.36	0.42	0.44	0.35	0.40	0.32	0.37	0.31	0.36	**0.28**	**0.35**
Human Activity	0.21	0.33	**0.05**	0.13	0.06	0.14	0.07	0.16	0.10	0.21	0.16	0.32	**0.05**	0.14	**0.05**	**0.12**
USHCN	0.30	0.35	0.18	0.26	0.18	0.25	0.36	0.38	0.25	0.30	0.19	0.26	**0.17**	**0.24**	0.19	0.25
CESNET	1.12	0.81	**0.98**	0.78	0.99	0.78	1.05	0.80	0.99	0.78	**0.98**	0.78	**0.98**	**0.77**	**0.98**	0.78
ClusterTrace	1.52	0.88	1.02	0.76	0.92	0.69	1.24	0.84	1.16	0.77	1.16	0.76	**0.89**	**0.68**	**0.89**	**0.68**
EPA-Air	3.00	1.19	0.34	0.46	0.36	0.46	1.77	0.86	0.54	0.59	0.55	0.59	**0.33**	**0.45**	**0.33**	**0.45**
FNSPID	0.22	0.33	**0.12**	**0.20**	0.13	0.22	0.13	0.22	0.19	0.30	0.23	0.36	**0.12**	0.21	**0.12**	**0.20**
GDELT	1.14	**0.69**	**1.03**	**0.69**	**1.03**	**0.69**	1.11	0.71	**1.03**	**0.69**	1.04	**0.69**	**1.03**	**0.69**	**1.03**	**0.69**
RepoHealth	1.08	0.62	0.60	0.42	0.63	0.46	0.65	0.42	1.18	0.76	0.67	0.51	0.61	0.44	**0.56**	**0.39**
StudentLife	1.33	0.86	0.76	0.64	0.76	**0.63**	0.97	0.70	0.76	**0.63**	0.75	0.64	**0.74**	**0.63**	0.76	**0.63**

Table 9: Zero-shot ablation results, following setup in Table 1.

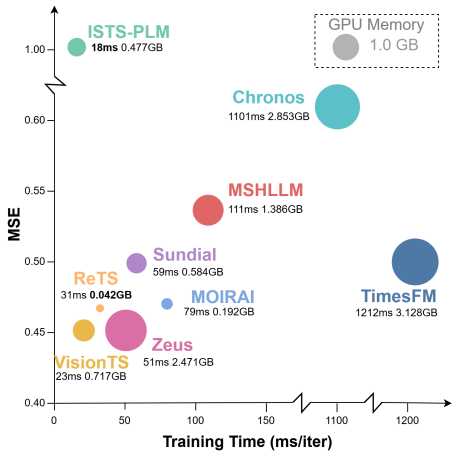
The detailed results of the ablation study are provided in Table 9. It should be noted that **w/o non-stationarity** did not match the performance of the full ReTS$_{\text{Luna}}$, due to the special non-stationarity root cause of Weather. As discussed in Appendix A.3, the non-stationarity in Weather is caused by the shifted mean of its observed values, while the std is relatively stable. Therefore, when **w/o non-stationarity** removes the whole normalization module, both mean and std normalizations are removed, which causes performance degradation. This is also the reason why ReTS$_{\text{Terra}}$ and ReTS$_{\text{Sol}}$ divide normalization operations into mean and std, realizing more fine-grained control and minor improvements on MTS datasets. What's more, since **w/o var** completely removes variable hyperedges and the attention modules, the hypergraph structure is no longer complete and affects the node-to-node message passing, causing its performance to degrade significantly on EPA-Air and ClusterTrace. To realize more fine-grained control over variable dependency learning, ReTS$_{\text{Terra}}$ and ReTS$_{\text{Sol}}$ can control the variable hyperedge message passing within the hypergraph attention module, without affecting the structure of the hypergraph. Moreover, the temporal and variable attention are split into



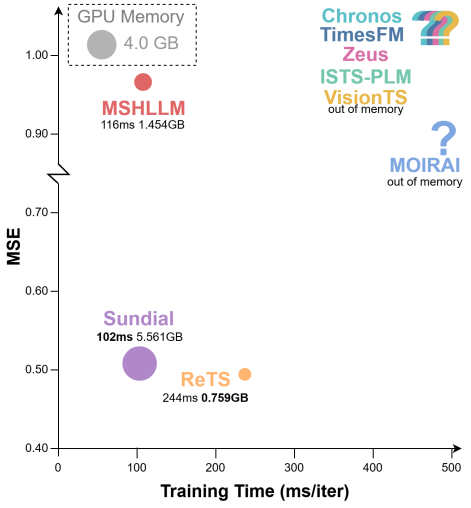
(a) ECL



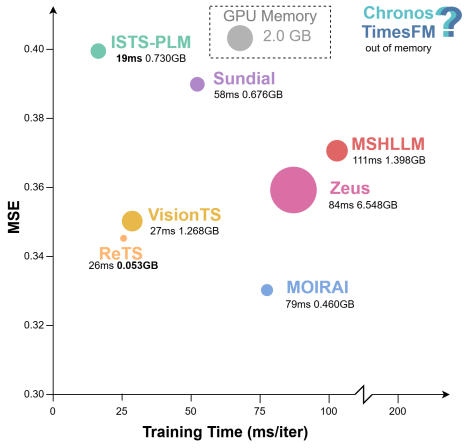
(b) ETTh1



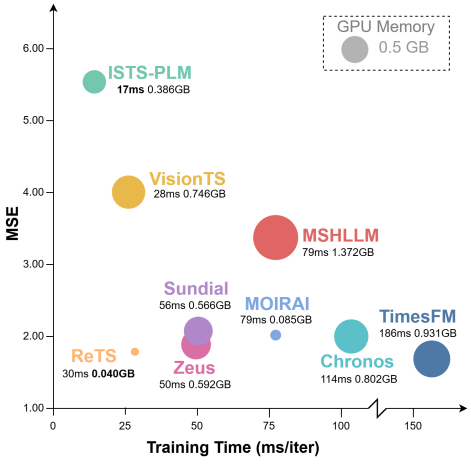
(c) ETTm1



(d) Traffic

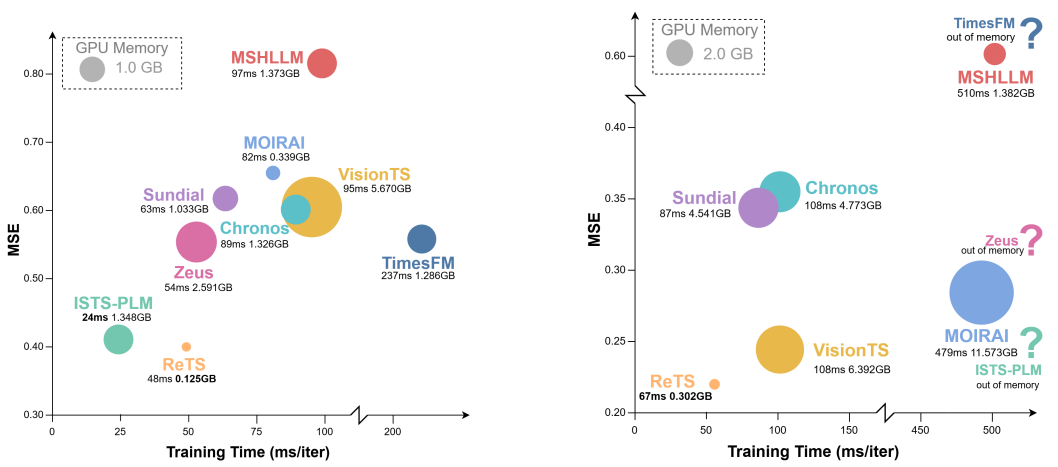


(e) Weather

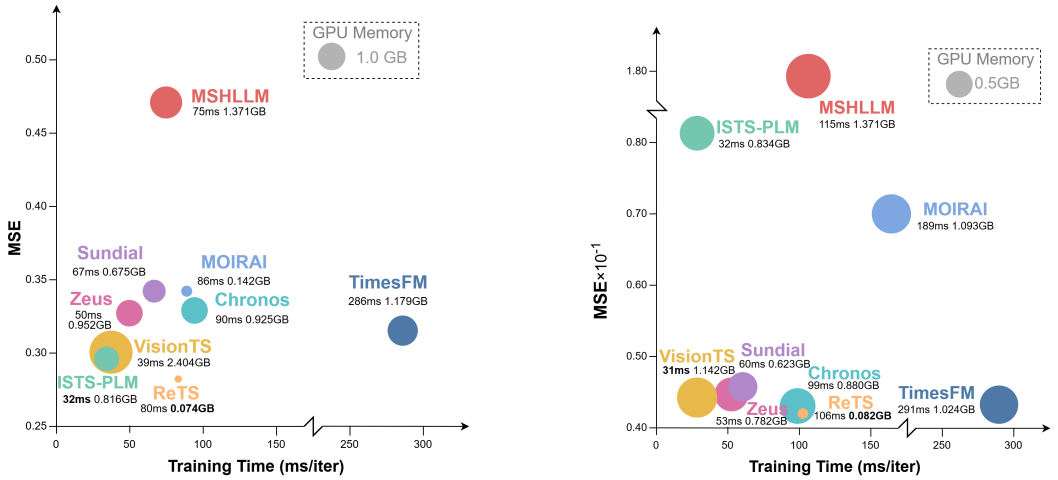


(f) ILI

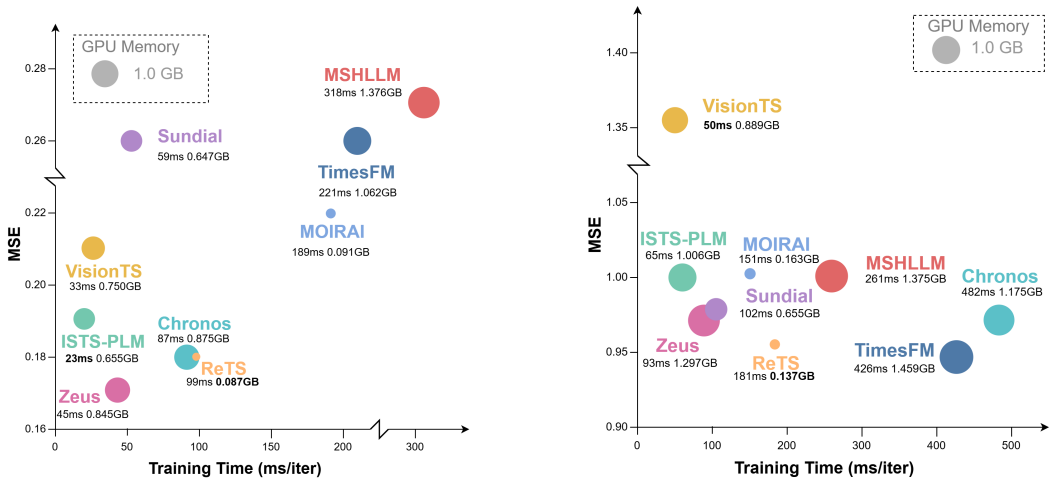
Figure 7: Full-shot training efficiency comparisons on MTS datasets with a batch size of 1. Models that cannot be fitted into 32 GB of GPU memory are marked as out of memory, as explained in Section D.



(a) MIMIC-III (b) MIMIC-IV



(c) PhysioNet'12 (d) Human Activity



(e) USHCN (f) CESNET

Figure 8: Full-shot training efficiency comparisons on six IMTS datasets with a batch size of 1, including MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, USHCN, and CESNET. Models that cannot be fitted into 32 GB of GPU memory are marked as out of memory, as explained in Section D.

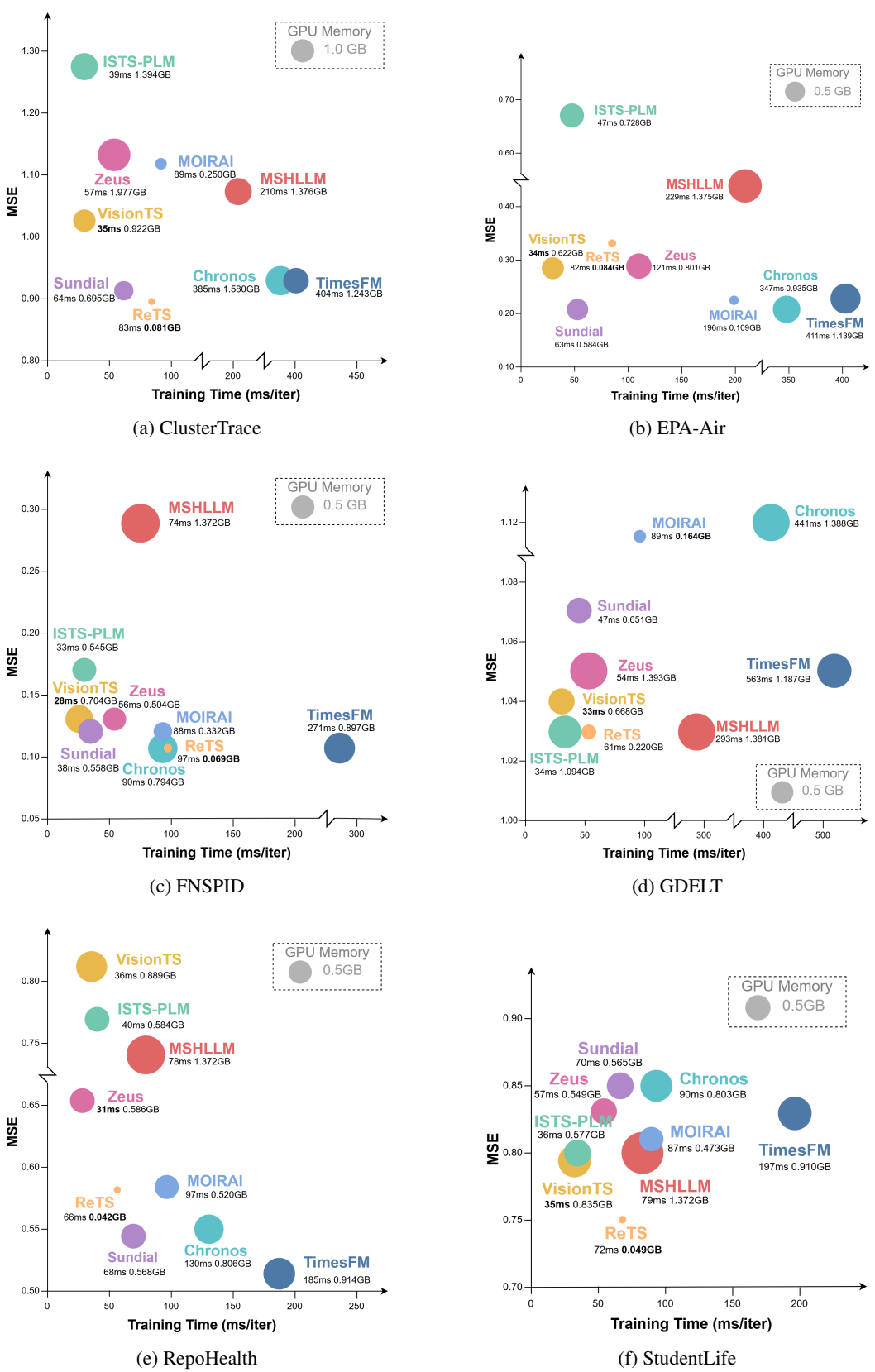


Figure 9: Full-shot training efficiency comparisons on six IMTS datasets with a batch size of 1, including ClusterTrace, EPA-Air, FNSPID, GDELT, RepoHealth, and StudentLife.

two modules. As for **w/o periodicity**, its performance is maintained on the GDELT dataset, where time periodicity is relatively weak, matching our motivation.

w/ NAS is the only ablation variant that introduces a different selection mechanism, and we describe its procedure here to make the comparison with $\text{ReTS}_{\text{Luna}}$ concrete. It keeps the same three-stage pre-training pipeline as $\text{ReTS}_{\text{Luna}}$, while replacing the property learner and the property-guided recurrent module activation with NAS in stage two. The search is conducted separately for each combination of the detected time series properties. For a given combination, an architecture is fully specified by the recurrence depth and, for each layer, the set of modules that are active. The full architecture, in which every eligible module is active, serves as the baseline candidate of the search space. A candidate architecture is drawn by keeping each eligible module at each of its allowed layers with probability 0.6. A candidate is feasible if and only if it produces a prediction and every stateful module completes its full activation pair. Infeasible candidates are discarded and redrawn, with the full architecture as the fallback. During the search, a pool of thirty-two candidate architectures is maintained for each property combination, and the candidate to train on each batch is chosen by an ϵ-greedy strategy, where ϵ decays linearly from 0.9 to 0.05 across the stage. Each candidate is fine-tuned for one hundred and twenty-eight steps before it is ever scored, so that a structure that needs adaptation is not discarded on its initial loss. Thereafter, the current best architecture is fine-tuned with gradients while every other selected candidate is evaluated without gradient computation, and candidates are ranked by the exponential moving average of their training MSE. The best architecture of each combination, in the sense of the lowest moving-average training MSE, is reused in stage three, and the searched architectures are applied at inference, in place of the property-guided recurrent module activation. The results of **w/ NAS** therefore correspond to a model whose module activation is determined by an explicit search budget, rather than by the learned activation of ReTS.

As for the ablation on pre-training process, we observe that ReTS can hardly be trained on target datasets without pre-training in advance, where the training process is unstable and the loss easily explodes. Therefore, the pre-training process helps ReTS begin finetuning from a safer starting point.

E.7 Contribution Discussions

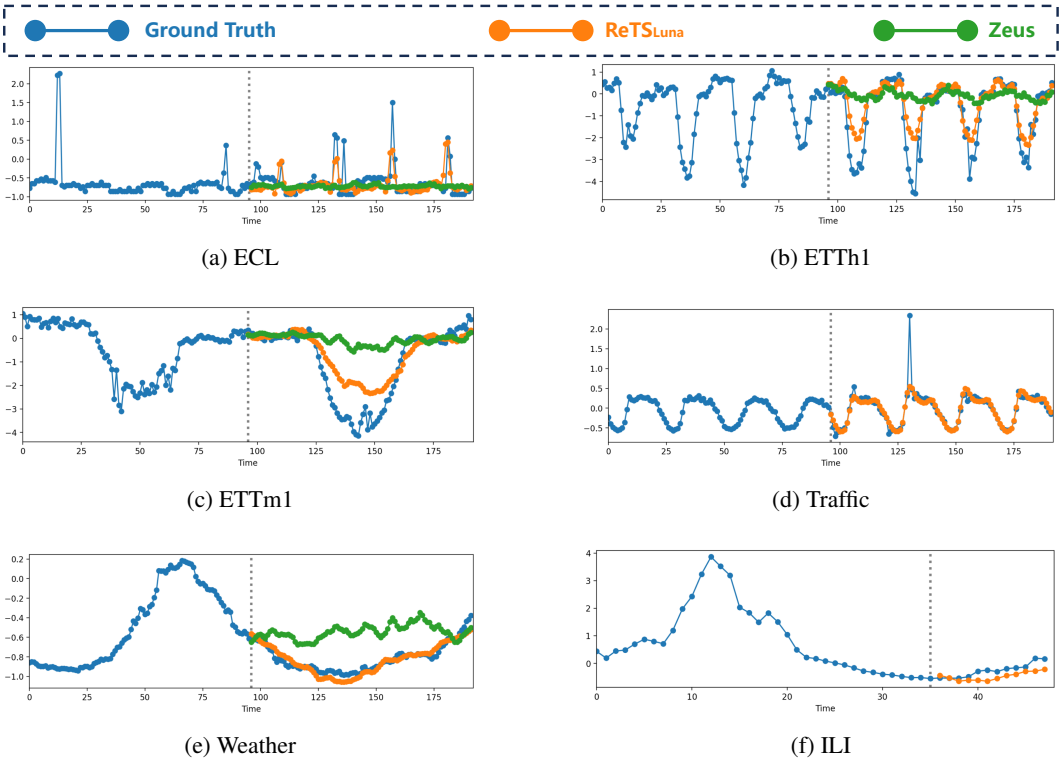
We additionally discuss the contributions of this paper. The most important contributions are our two-fold insights on the time series forecasting task: (1) providing a high-level time series property view on existing methods, instead of focusing on the low-level code implementation view, such that many of them can be re-expressed using a unified recurrent structure in Figure 2. This can help researchers understand the potential similarities and differences among the numerous existing methods. (2) providing a time series property matching approach to build time series foundation models, apart from the parameter scale-up approach following LLMs. Larger parameters in time series foundation models may not necessarily bring better accuracy, and matching the parameters in modules with their suitable time series input can be a new approach.

As for ReTS, it is built to demonstrate the potential of the above insights. While ReTS is built upon some already verified mechanisms, its uniqueness lies in bringing these insights into its model architectural design. The ”everything as modules” overall structure and the property related designs show potential in dynamically transforming the model structure at run time based on the input time series.

F Visualization

F.1 Forecasting Results

We visualize the zero-shot forecasting results of ReTS and Zeus (Fu et al., 2026), as depicted in Figures 10, 11, and 12. Ground truth is in blue, ReTS in orange, and Zeus in green. Since the patch-based baselines produce incompatible tensor shapes on IMTS datasets and cannot be placed together in one figure, we compare against Zeus, a non-patch-based baseline. The forecast settings are exactly the same as in Table 1. Since Zeus is excluded on Traffic, ILI, USHCN, and ClusterTrace in Table 1, it is omitted from the corresponding visualizations. As shown in Figures 10–12, ReTS is closer to the ground truth value for most of the datasets.

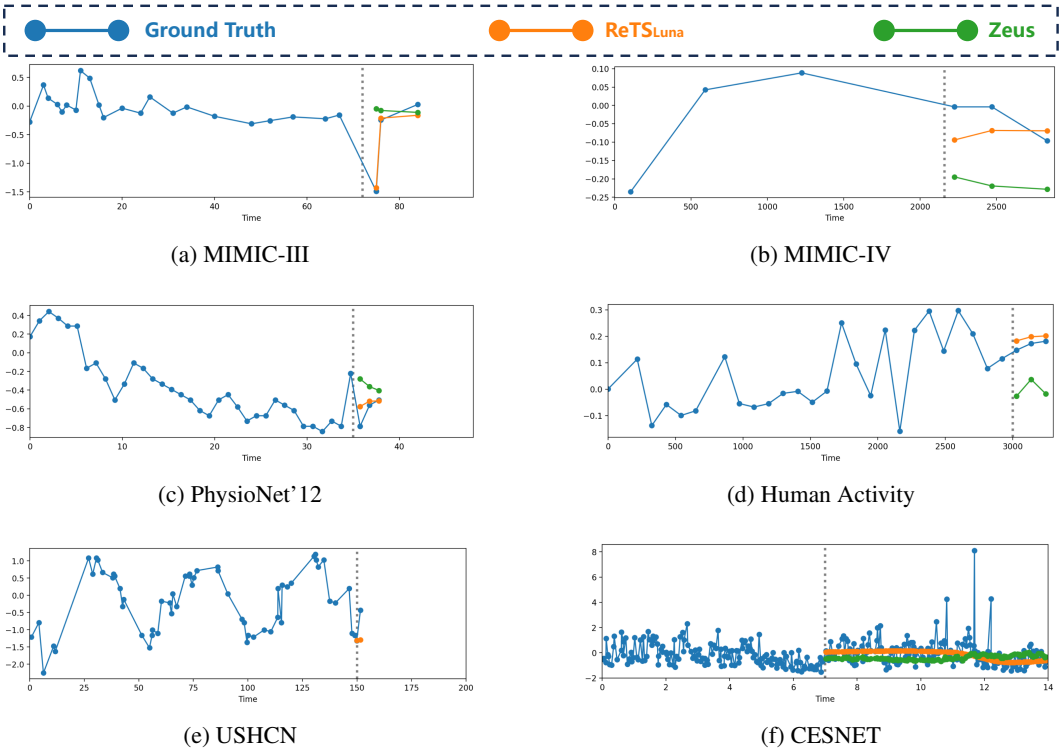


(a) ECL (b) ETTh1

(c) ETTm1 (d) Traffic

(e) Weather (f) ILI

Figure 10: Zero-shot forecasting visualizations of ReTS and Zeus on MTS datasets. Ground truth is in blue, ReTS in orange, and Zeus in green. The vertical dashed line indicates the boundary between the lookback window and the forecast horizon. Zeus is omitted on Traffic and ILI, as it is excluded on these datasets in Table 1. The forecast horizon used in the visualization for ILI is 12, and 96 for the rest of the datasets.

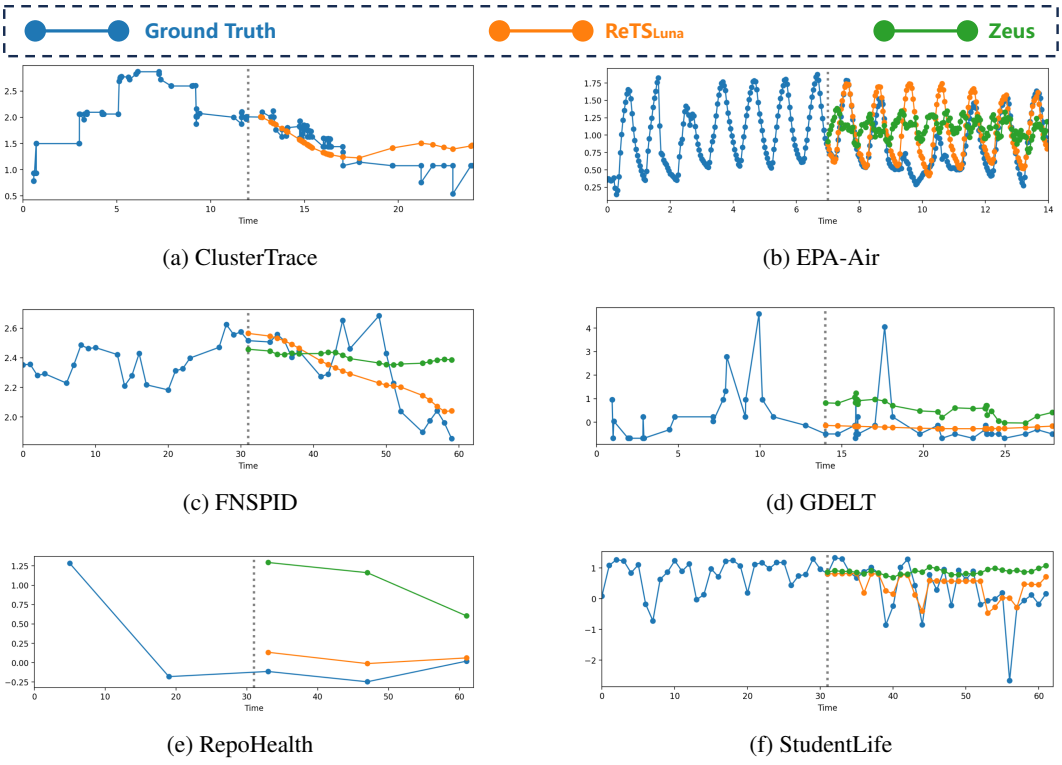


(a) MIMIC-III (b) MIMIC-IV

(c) PhysioNet'12 (d) Human Activity

(e) USHCN (f) CESNET

Figure 11: Zero-shot forecasting visualizations of ReTS and Zeus on six IMTS datasets, including MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, USHCN, and CESNET. Ground truth is in blue, ReTS in orange, and Zeus in green. The vertical dashed line indicates the boundary between the lookback window and the forecast horizon. Zeus is omitted on USHCN, as it is excluded on this dataset in Table 1.



(a) ClusterTrace (b) EPA-Air

(c) FNSPID (d) GDELT

(e) RepoHealth (f) StudentLife

Figure 12: Zero-shot forecasting visualizations of ReTS and Zeus on six IMTS datasets, including ClusterTrace, EPA-Air, FNSPID, GDELT, RepoHealth, and StudentLife. Ground truth is in blue, ReTS in orange, and Zeus in green. The vertical dashed line indicates the boundary between the lookback window and the forecast horizon. Zeus is omitted on ClusterTrace, as it is excluded on this dataset in Table 1.

G BASELINE DETAILS

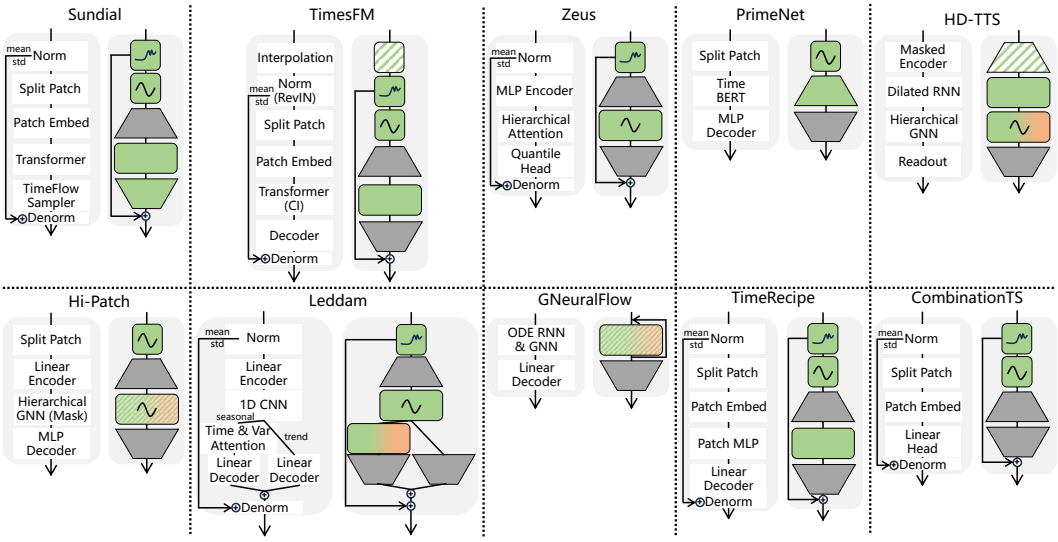


Figure 13: Additional decomposition of forecasting baselines. MSHLLM, ISTS-PLM, VisionTS, and Chronos are based on models from NLP or CV, mTAN is mainly designed for the irregular time series classification task, so they are not illustrated here.

We briefly introduce each baseline model along with its key hyperparameter settings. Also, an additional decomposition of the forecasting baselines is available in Figure 13. It should be noted that some operations upon timestamp or mask inputs, like positional encoding, are not visualized here, since our motivation mainly discusses the properties upon the observed values of time series inputs. Also, tensor reshape operations are not detailed in the figures.

Hyperparameter search aims to minimize the loss on the validation sets. The number of layers is searched within 1, 2, 3, and 4, and the hidden dimension size is searched within 16, 32, 64, 128, 256, and 512. Unless otherwise specified, we use a batch size of 16 for USHCN, and 32 for others. However, if the model cannot be trained using 80 GB of GPU memory, we recursively halve the batch size until training is possible. The number of epochs, early stopping patience, and learning rates have been described in Section D. We aim to use the same hyperparameters for learning rate and special loss functions, as specified in their original papers and codes, whenever available.

G.1 PRE-TRAINED METHODS

MOIRAI (Woo et al., 2024) is a pre-training model for time series forecasting. We use the small variant of the provided pre-trained configurations and weights following the settings in previous benchmarking (Li et al., 2025), which comprises 6 layers with hidden dimension 384. We fine-tune the model on IMTS datasets with a learning rate of 1×10^{-4}.

MSHLLM (Shang et al., 2026) aligns large language models with time series through multi-scale hypergraph representations. A hyperedging mechanism enhances multi-scale semantic information of the time series semantic space, and a cross-modality alignment module bridges the modality gap between natural language and time series at different scales. A mixture of prompts mechanism provides contextual information to help the LLM understand multi-scale temporal patterns. The model is adapted for forecasting by treating time series as token sequences and predicting future values through the aligned LLM.

ISTS-PLM (Zhang et al., 2025) adapts pre-trained language models for irregularly sampled time series analysis. It represents irregular time series as structured sequences with explicit time intervals

and variable identifiers, enabling PLMs to model both intra-series temporal dependencies and inter-series variable correlations. Time-aware and variable-aware attention mechanisms are integrated to handle uneven sampling intervals and prevalent missing data.

Sundial (Liu et al., 2025c) is a family of native time series foundation models that predict the next-patch distribution with a flow-matching objective, so it operates on continuous values without discrete tokenization. Since Sundial Base is the only variant that is publicly available, we use it during our benchmark comparisons.

VisionTS (Chen et al., 2025) repurposes a visual masked autoencoder pre-trained on ImageNet as a time series forecaster. It reformulates time series forecasting as an image reconstruction task by reshaping time series data into 2D image-like representations, leveraging the intrinsic similarities between natural images and real-world time series. No domain-specific adaptation is required; the model operates in zero-shot mode using the ImageNet-pre-trained weights directly.

Chronos (Ansari et al., 2024) tokenizes time series values into a fixed vocabulary and trains transformer-based language models on the tokenized series with a cross-entropy loss. We use the Chronos-Bolt variant in our comparison, following the comparison in Zeus (Fu et al., 2026). We use the provided pre-trained weights in zero-shot mode.

TimesFM (Das et al., 2024) pre-trains a decoder-style attention model with input patching on a large corpus of real and synthetic time series. We choose the TimesFM 2.5 variant in our comparison, since it is the latest and strongest variant in the TimesFM family by the time this manuscript is being prepared. We use the provided pre-trained weights in zero-shot mode.

Zeus (Fu et al., 2026) is a tuning-free foundation model that uses a multi-scale Transformer with point-wise tokenization and a U-shaped hierarchy.

PrimeNet (Chowdhury et al., 2023) is an IMTS pre-training model. However, since its open-sourced weights are specific to datasets with 41 variables, we retrain the model from scratch on all datasets, and it is therefore placed in Table 8. The patch lengths are 12, 180, 6, 300, and 10 for MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN respectively. The number of heads in attention is 1. The learning rate is 1×10^{-4}. The hidden dimension size is set to 32, 32, 128, 128, and 64 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. On the six regular datasets (ECL, ETTh1, ETTm1, Traffic, Weather, and ILI), the patch length is 3 and the hidden dimension size is 128. On the seven remaining IMTS datasets (CESNET, ClusterTrace, EPA-Air, FNSPID, GDELT, RepoHealth, and StudentLife), the patch length is 31 and the hidden dimension size is 128.

G.2 Task-specific Methods

HD-TTS (Marisca et al., 2024) implements both variable and temporal multi-scale inside the backbone. The number of RNN layers is 4. The number of pooling layers is 1. The learning rate is 1×10^{-3}. The hidden dimension size is 64. These settings are used on all eighteen datasets.

Hi-Patch (Luo et al., 2025) implements hierarchical graph fusion inside the backbone. The patch lengths are 12, 90, 6, 500, and 6 for MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. The number of attention heads is 8 on the USHCN dataset and 1 on the rest. The learning rate is 1×10^{-3}. The number of layers is set to 4, 1, 4, 4, and 4 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. The hidden dimension size is set to 32, 64, 32, 128, and 32 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. On the six regular datasets (ECL, ETTh1, ETTm1, Traffic, Weather, and ILI), the patch length is 6, the number of layers is 1, the number of attention heads is 1, and the hidden dimension size is 64. On the seven remaining IMTS datasets (CESNET, ClusterTrace, EPA-Air, FNSPID, GDELT, RepoHealth, and StudentLife), the patch length is 31, the number of layers is 4, the number of attention heads is 1, and the hidden dimension size is 32.

HyperIMTS (Li et al., 2025) represents observed values as nodes in a hypergraph, interconnected by temporal and variable hyperedges for irregularity-aware message passing. The number of layers is set to 2, 4, 1, 3, and 1 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. The number of attention heads is set to 4, 8, 8, 1, and 1 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. The hidden dimension size is set to 256, 128, 256, 128, and 256 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. The learning rate is 1×10^{-3}. On the six regular datasets (ECL, ETTh1, ETTm1, Traffic, Weather, and ILI) and the seven remaining IMTS datasets (CESNET, ClusterTrace, EPA-Air, FNSPID, GDELT, RepoHealth, and StudentLife), the number of layers is 2, the number of attention heads is 4, and the hidden dimension size is 128.

mTAN (Shukla & Marlin, 2021) converts IMTS to reference points for IMTS classification. The number of reference points is 32 on the MIMIC-III dataset, 128 on the USHCN dataset, and 8 on the rest of the datasets. The hidden dimension size is set to 32, 64, 64, 64, and 256 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. The learning rate is 1×10^{-3}. On the six regular datasets, the reference points are 8 and the hidden dimension size is 512. On the seven remaining IMTS datasets, the reference points are 8 and the hidden dimension size is 64.

FreqMoE (Liu, 2025) transforms the input into the frequency domain with the fast Fourier transform, splits the frequency components into three bands, and processes each band with a specialized expert network whose outputs are combined by a gating mechanism on the frequency magnitudes. The aggregated representation is then passed through a stack of frequency-domain prediction blocks with residual connections. The number of prediction blocks is 3 and the number of experts is 3. The dropout rate is 0.3. The learning rate is 1×10^{-3}. These settings are used on all datasets.

Leddam (Yu et al., 2024) uses learnable seasonal-trend decomposition for MTS forecasting. The learning rate is 1×10^{-3}. The number of layers is set to 2, 4, 2, 1, and 4 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. The hidden dimension size is set to 32, 32, 128, 512, and 64 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. On the six regular datasets, the number of layers is 1 and the hidden dimension size is 512. On the seven remaining IMTS datasets (CESNET, ClusterTrace, EPA-Air, FNSPID, GDELT, RepoHealth, and StudentLife), the number of layers is 2 and the hidden dimension size is 128.

PatchTST (Nie et al., 2023) leverages patching in the transformer for MTS forecasting. The patch lengths are 12, 90, 6, 300 and 10 for MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. The number of encoder layers is 8 on the USHCN dataset, and 3 on the rest. The attention factor is 3. The number of heads in attention is 8 on the USHCN dataset, and 16 on the rest. The learning rate is 1×10^{-4}. The hidden dimension size is set to 128, 128, 256, 512, and 32 on the datasets MIMIC-III, MIMIC-IV, PhysioNet'12, Human Activity, and USHCN, respectively. On the six regular datasets, the patch length is 16 for ECL, ETTh1, ETTm1, Traffic, and Weather, and 18 for ILI, and the hidden dimension size is 512. The number of encoder layers is 2 for ECL, Traffic, and Weather, 1 for ETTh1, 4 for ILI, and 1, 3, 1, and 3 for ETTm1 at horizons 96, 192, 336, and 720, respectively. The number of heads in attention is 8 for ECL, 16 for Traffic, and 4 for ILI. For the remaining three datasets, the number of heads in attention varies with the forecast horizon. It is 2, 8, 8, and 16 for ETTh1, 2, 8, 4, and 4 for ETTm1, and 4, 16, 4, and 4 for Weather at horizons 96, 192, 336, and 720, respectively. On the seven remaining IMTS datasets (CESNET, ClusterTrace, EPA-Air, FNSPID, GDELT, RepoHealth, and StudentLife), the patch length is 31, the number of encoder layers is 3, the number of heads in attention is 16, and the hidden dimension size is 256.

GNeuralFlow (Mercatali et al., 2024) enhances NeuralFlows with graph neural networks for IMTS analysis. The flow model uses ResNet. The number of flow layers is 2. The latent dimension for input is 20. The latent dimension for time is 8. The number of hidden layers is 3. The learning rate is 1×10^{-3}. These settings are used on all eighteen datasets.

TimeRecipe (Zhao et al., 2026) is a modular benchmarking framework that evaluates individual design components across datasets and forecasting horizons. It provides empirical recommendations

Table 10: Summary of all eighteen time series datasets. Avg # obs. (padding) refers to the averaged number of observations after applying canonical zero-padding on IMTS. Therefore, as regular datasets are fully observed, Avg # obs. and Avg # obs. (padding) do not apply to them. Since regular time series datasets use sliding window upon raw series to obtain samples in datasets, their # Sample depends on the length of window size and sliding stride, marked as N/A.

Dataset	Max length	# Variable	# Sample	# Series	Avg # obs.	Avg # obs. (padding)
MIMIC-III	96	96	21,250	21,250	144.6	9,216.0
MIMIC-IV	971	100	17,874	17,874	304.8	92,000.0
PhysioNet'12	47	36	11,981	11,981	308.6	1,692.0
Human Activity	131	12	1,359	25	362.2	1,573.2
USHCN	337	5	1,114	1,114	313.5	1,685.0
ETTh1	17,520	7	N/A	1	N/A	N/A
ETTm1	70,080	7	N/A	1	N/A	N/A
ECL	17,520	321	N/A	1	N/A	N/A
Traffic	17,534	862	N/A	1	N/A	N/A
Weather	35,040	11	N/A	1,603	N/A	N/A
ILI	1,040	1	N/A	1	N/A	N/A
CESNET	51,107	10	46	11	17,092.0	51,107.0
ClusterTrace	12,615	11	342	3	23,000.3	12,615.0
EPA-Air	6,587	4	202	8	6,194.0	6,587.0
FNSPID	3,659	6	1,586	10	20,968.8	3,659.0
GDELT	34,317	5	808	8	24,150.6	34,317.0
RepoHealth	6,783	10	393	4	16,957.5	6,783.0
StudentLife	1,743	9	828	20	7,680.5	1,743.0

for architecture design based on over 10,000 experiments. We use the recommended component combinations for each dataset.

CombinationTS (Wang et al., 2026) decomposes forecasting models into orthogonal modules and evaluates each component under a shared evaluation condition space. It identifies which components truly drive performance through marginalized performance and stability analysis. We use the optimal component combinations identified by the framework.

H Dataset Details

We first introduce the eighteen datasets used in benchmarks. MIMIC-III (Johnson et al., 2016) is a clinical database collected from ICU patients during the initial 48 hours of admission, aggregated to 30-minute intervals. MIMIC-IV (Johnson et al., 2023) is built upon MIMIC-III, with a higher sampling frequency and data aggregated to 1-minute intervals. PhysioNet'12 (Silva et al., 2012) is also a clinical database collected during the first 48 hours of ICU stay, aggregated to 1-hour intervals. Human Activity includes biomechanical data detailing 3D positional variables, aggregated to 1-millisecond intervals. USHCN (Menne et al., 2016) includes climate data spanning over 150 years, collected from meteorological stations distributed across the United States. Our analysis focuses on a subset of 4 years between 1996 and 2000. ECL (Zhou et al., 2021) collects hourly electricity consumption of 321 customers from 2012 to 2014. ETT (Zhou et al., 2021) contains electricity transformer temperature data from two counties in China, with ETTh1 recorded hourly and ETTm1 recorded at 15-minute intervals, each comprising oil temperature and six power load features. Traffic (Wu et al., 2021) describes road occupancy rates measured by sensors on San Francisco Bay area freeways, collected hourly. Weather (Wu et al., 2021) contains 21 meteorological indicators, such as air temperature and humidity, recorded every 10 minutes throughout 2020. ILI (Wu et al., 2021) includes weekly influenza-like illness patient data from the CDC between 2002 and 2021, describing the ratio of ILI patients to total patients. CESNET (Chang et al., 2025) captures network flow records exhibiting scheduling jitter due to internal system constraints. ClusterTrace (Chang et al., 2025) records GPU cluster telemetry with resource-aware sampling, yielding structured but dynamic irregularity. EPA-Air (Chang et al., 2025) compiles data from heterogeneous outdoor air quality sensors reporting at unsynchronized intervals. FNSPID (Chang et al., 2025) contains financial time series sampled only during market operating hours. GDELT (Chang et al., 2025) corresponds

Table 11: The fraction of samples in each dataset that expose each time series property, in percentage. The properties are detected with the property detectors of ReTS.

Dataset	Periodic	Non-stationary	Missing	Variable dependency
MIMIC-III	24.3%	18.6%	100.0%	85.4%
MIMIC-IV	45.8%	36.1%	100.0%	78.0%
PhysioNet'12	75.3%	90.3%	100.0%	85.4%
Human Activity	77.9%	95.6%	100.0%	19.2%
USHCN	98.3%	71.5%	100.0%	22.8%
ETTh1	99.8%	16.3%	0.0%	100.0%
ETTm1	99.3%	98.6%	0.0%	99.7%
ECL	100.0%	0.0%	0.0%	100.0%
Traffic	100.0%	0.1%	0.0%	100.0%
Weather	100.0%	98.2%	0.0%	100.0%
ILI	100.0%	94.0%	0.0%	100.0%
CESNET	0.0%	87.0%	100.0%	100.0%
ClusterTrace	23.4%	69.6%	100.0%	19.0%
EPA-Air	62.9%	68.3%	100.0%	8.9%
FNSPID	77.2%	94.1%	100.0%	100.0%
GDELT	20.5%	39.1%	100.0%	59.0%
RepoHealth	57.8%	33.1%	66.4%	100.0%
StudentLife	19.8%	65.1%	100.0%	37.1%

to discrete global events such as political crises or major policy changes. RepoHealth (Chang et al., 2025) reflects GitHub project histories with adaptive sampling, denser during high commit activity. StudentLife (Chang et al., 2025) includes smartphone sensing data with irregularity shaped by human activity patterns.

We then cover the lookback windows and forecast horizons used in benchmarking. For MTS datasets, the lookback windows are 96 steps for ECL, ETTh1, Traffic, and Weather, 24 steps for ETTm1, and 36 steps for ILI. For IMTS datasets, the lookback windows are 36 hours for MIMIC-III, MIMIC-IV, and PhysioNet'12, 3000 milliseconds for Human Activity, and 3 years for USHCN. For the rest IMTS datasets, the lookback windows are 7 days for CESNET and EPA-Air, 12 hours for ClusterTrace, 14 days for GDELT, and 31 days for FNSPID, RepoHealth, and StudentLife. As for the forecast horizon, ECL, ETTh1, ETTm1, Traffic, and Weather cover $\{96, 192, 336, 720\}$, and ILI covers $\{12, 24, 36, 48\}$. Human Activity uses a forecast horizon of 300 milliseconds. For MIMIC-III, MIMIC-IV, PhysioNet'12, and USHCN, the forecast horizon is the next 3 timestamps. For the rest IMTS datasets, the forecast horizons are 7 days for CESNET and EPA-Air, 12 hours for ClusterTrace, 14 days for GDELT, and 31 days for FNSPID, RepoHealth, and StudentLife.

We further characterize the eighteen datasets by the time series properties exposed by their samples, summarized in Table 11. Each sample is probed with the property detectors of ReTS on exactly the tensors the model receives, and a sample is counted for a property when the corresponding detector signals it. These detectors are the same ones that drive the property-guided recurrent module activation of ReTS at inference time: time periodicity by a spectral concentration score, time non-stationarity by the Augmented Dickey-Fuller unit-root test, time missingness by the completeness of the observation mask, and variable dependencies by the eigenvalue effective dimensionality of the pairwise-complete correlation matrix. The fractions vary substantially across datasets and, in most cases, across properties within a dataset, so that no single dataset exposes only a single property. In particular, the regular datasets (ETTh1, ETTm1, ECL, Traffic, Weather, and ILI) are fully observed and therefore show no time missingness, whereas the irregular datasets are at least partially missing. Among the regular datasets, ECL, Traffic, and ETTh1 are nearly stationary, while ETTm1, Weather, and ILI are largely non-stationary, and CESNET and a few irregular datasets are non-periodic. The strong dependence of the variable dependency fraction on the dataset, from 8.9% on EPA-Air to

100% on several datasets, indicates that the cross-variable redundancy captured by the detector is dataset specific.

We provide the information about the eighteen datasets used during benchmarking in Table 10, which are also the pre-training datasets of ReTS. It should be noted that we adopt the leave-one-out strategy during pre-training when evaluating the zero-shot and few-shot performance of ReTS. That is, if ECL is the dataset to be evaluated, we will drop ECL from the pre-training dataset for fair comparison. It should also be noted that the Time-IMM overview table (Chang et al., 2025) reports thirty entities for CESNET, but its description in the main paper content reports "We include 11 entities in total". We investigate this discrepancy and confirm that the released data contains only eleven entities, which is the count we report in Table 10.

The train/val/test split follows their corresponding standard practices. For the five irregular datasets, the split is 80/10/10 by sample count, following HyperIMTS (Li et al., 2025). For the six regular datasets, the split is time-based and chronological: twelve, four, and four months for ETTh1 and ETTm1; fifteen, three, and four months for ECL; twenty-eight, ten, and ten months for Weather; and 70/10/20% for Traffic and ILI, following Time-Series-Library (Wang et al., 2024b). For CESNET, ClusterTrace, EPA-Air, FNSPID, GDELT, RepoHealth, and StudentLife, the split is 60/20/20 and applied chronologically within each entity, following Time-IMM (Chang et al., 2025).