SVTime: Small Time Series Forecasting Models Informed by "Physics" of Large Vision Model Forecasters

ChengAo Shen[1], Ziming Zhao[1], Hanghang Tong[2], Dongjin Song[3], Dongsheng Luo[4],
Qingsong Wen[5], Jingchao Ni[1]

[1]University of Houston, [2]University of Illinois at Urbana-Champaign, [3]University of Connecticut,
[4]Florida International University, [5]Squirrel Ai Learning

[1]{cshen9,zzhao35,jni7}@uh.edu, [2]htong@illinois.edu, [3]dongjin.song@uconn.edu,
[4]dluo@fiu.edu, [5]qingsongedu@gmail.com

Abstract

Time series AI is crucial for analyzing dynamic web content, driving a surge of pre-trained large models known for their strong knowledge encoding and transfer capabilities across diverse tasks. However, given their energy-intensive training, inference, and hardware demands, using large models as a one-fits-all solution raises serious concerns about carbon footprint and sustainability. For a specific task, a compact yet specialized, high-performing model may be more practical and affordable, especially for resource-constrained users such as small businesses. This motivates the question: Can we build cost-effective lightweight models with large-model-like performance on core tasks such as forecasting? This paper addresses this question by introducing SVTime, a novel Small model inspired by large Vision model (LVM) forecasters for long-term Time series forecasting (LTSF). Recently, LVMs have been shown as powerful tools for LTSF. We identify a set of key inductive biases of LVM forecasters — analogous to the "physics" governing their behaviors in LTSF — and design small models that encode these biases through meticulously crafted linear layers and constraint functions. Across 21 baselines spanning lightweight, complex, and pre-trained large models on 8 benchmark datasets, SVTime outperforms state-of-the-art (SOTA) lightweight models and rivals large models with $10^3\times$ fewer parameters than LVMs, while enabling efficient training and inference in low-resource settings.

1 Introduction

The World Wide Web is a dynamic, ever-evolving system that continuously produce time series data pertaining to web traffic (*e.g.*, page views), user behavior (*e.g.*, bounce rates), web content (*e.g.*, trending topics), e-commerce (*e.g.*, click-through rates), system security (*e.g.*, latency logs), and so on, where the ability to anticipate and respond to changing patterns and user behaviors plays a crucial role. As such, time series forecasting — analyzing historical data and predicting future trends — emerges as an indispensible component of modern web technologies [15, 17, 19, 22, 46], driving intelligent web services such as content recommendation [42], microservice monitoring [20], and web economics modeling [46].

Inspired by the success of large models in AI and their strong adaptability across modalities, emergent methods for time series forecasting have explored Transformer [29, 32, 45, 53, 54], Time Series Foundation Models (TSFMs) [1, 6, 9, 41, 43], Large Language Models (LLMs) [21, 27, 33, 55], Large Vision Models (LVMs) [4, 34, 36] and Vision-Language Models (VLMs) [52]. However, given their energy-intensive training, inference, and hardware demands, using large models as a one-fits-all solution raises serious concerns about carbon footprint and sustainability [3]. For a specific task, recent findings reveal that most of a large model's parameters may be useless [30]. In language domain, small language models (SLMs) such as Microsoft Phi series [16], NVIDIA Hymba [7], and DeepSeek-R1-Distill series [11] are becoming on par with LLMs on certain tasks, powering economical development of agent systems [2]. The trend toward compact yet high-performing models specialized in core tasks, along with the need for quick deployment in resource-constrained scenarios (*e.g.*, edge devices, small institutes and businesses), motivate the question: Can we build cost-effective models with large-model-like performance on core tasks such as long-term time series forecasting (LTSF)?

This question is challenging due to the trade-off between model capacity and efficiency. Large models — *e.g.*, with millions or billions of parameters — support complex attention mechanisms, and encoding of knowledge by large-scale pre-training. In contrast, smaller models are restricted in design possibilities and unfit to pre-training because of underfitting, thus appear to be less likely to rival (pre-trained) large models. A straightforward direction is to explore knowledge-distillation (KD), such as distilling OccamVTS from LVMs [30] and TimeDistill from Transformers [31]. Whereas, KD is still resource-demanding as it relies on the large teacher models, which will be loaded and communicated with during the training of student models. Also, these student models are non-competitive in reducing parameters as evaluated in §4.4.

Recent efforts toward small models for LTSF focus on exploring different hypotheses pertaining to the LTSF task, such as point-wise correlation [50], segment-wise correlation [38], periodicity [24, 25], and time-frequency relationships [47, 50], while designing
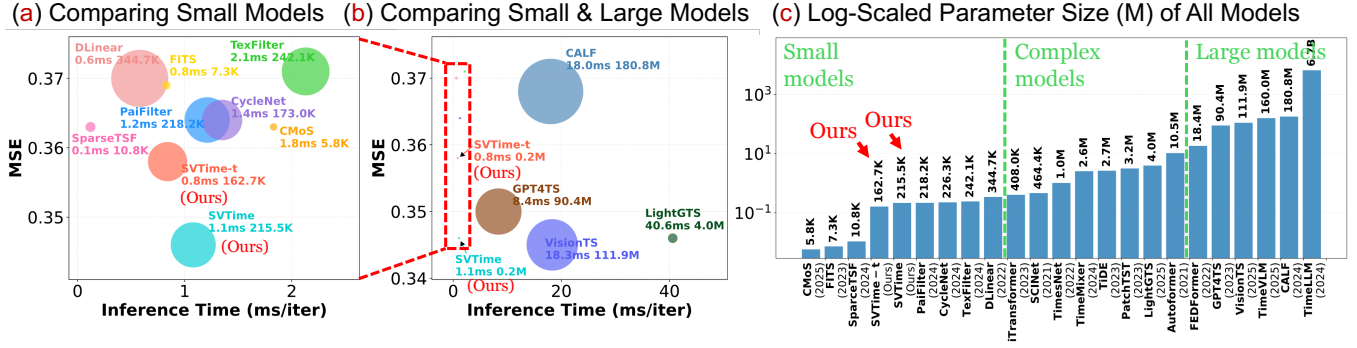
Figure 1: An overview of (a) forecasting performance vs. inference time on ETTm1 dataset, where circle size reflects model size, which is a zoom-in of the small models in (b); in (b), `TimeLLM` [21] is ignored for its much larger size and longer inference time (see §4.2); and (c) is a categorization of small, complex, and pre-trained large models.

models with inductive biases encoding the hypotheses. For example, `SparseTSF` [25] assumes inter-period smoothness of time series and forecasts future periods by aggregating past periods. `CycleNet` [24] also utilizes periods but makes them learnable in a seasonal-trend-like framework. `CMoS` [38] extends `DLinear` [50] and assumes the correlation between historical chunks (*i.e.*, a segment of time series) and future chunks can be modeled by linear layers. However, as the hypotheses don't align with large models, none of these models is comparable to pre-trained large models (as evaluated in §4).

In this paper, in addition to centering on the LTSF task itself, we take a new perspective by grounding our hypotheses on the behavior of large models. We analyze a specific LVM, `MAE` [13] — when used as a forecaster in `VisionTS` [4] — for its superior performance in LTSF over LLMs and VLMs as validated by [35, 51]. Our analysis uncovers a set of key inductive biases — analogous to the "physics" governing `MAE`'s behavior in LTSF — including (1) inter-period consistency (§3.1); (2) patch-wise variety (§3.2); and (3) distance-attenuating local attention (§3.3). We design lightweight models with linear layers and constraint functions to encode these biases, sharing similar merits with physics-informed learning [14, 18], which is useful in learning cost-effective models. Using the first two "physics", we propose SVTime, a novel Small model inspired by LVM for long-term Time series forecasting. Including the third "physics" leads to a tiny version of the model, namely SVTime-t. Moreover, to complement the biases toward forecasting periods, we encapsulate our models within a backcast-residual based decomposition framework (§3.4), which adaptively compensates forecasts with residual trends. As Fig. 1(a)(b) illustrate, with only 0.2% (0.1%) size of `VisionTS`, SVTime (SVTime-t) shows superiority in the small model regime and rivals pre-trained large models including LLMs, LVMs, TSFMs and VLMs. To sum up, our contributions are as follows.

- **Discovery**. We dive in to the behavioral patterns of a SOTA LVM forecaster, uncover key inductive biases, and validate their value by explicitly transferring them to a much simpler model.
- **Development**. We carefully design lightweight models using linear layers and constraint functions to reproduce the "physics" of an LVM forecaster within a parameter-limited regime.
- **Evaluation**. We compare SVTime(-t) with 21 SOTA baselines covering lightweight, complex, and pre-trained large models on

8 benchmark datasets, along with extensive ablations, validating their small-model-like sizes and large-model-like performance.

Categorization of Models. In this paper, we categorize models as *lightweight*, *complex*, and *pre-trained large* models according to our observation of the 21 SOTA models as shown in Fig. 1(c). `DLinear` is used to separate lightweight and complex models as the smaller models mostly consist of linear layers, while the larger models employ CNN, MLP, or Transformer with more complex designs. Pre-trained large models only include models that have been pre-trained on some datasets. However, this categorization is subject to discussion and has little bearing on the essence of the paper.

2 Related Work

To the best of our knowledge, this is the first work to explore LVMs' inductive biases for guiding the design of lightweight forecasting models. Our work relates to **Large Models for time series forecasting (TSF)**, **Lightweight models for TSF**, and **Knowledge-Distillation (KD)-based TSF**, which are discussed below.

Large Models for TSF. Recent research on TSF draws a lot of attention to Transformer [29, 32, 45, 53, 54], TSFMs [1, 6, 9, 41, 43], LLMs [21, 27, 33, 55], LVMs [4, 34, 36], and multimodal models [35, 52]. Early Transformer-based forecasters, such as `Informer` [53] and `Autoformer` [45], focus on encoding time points. More recent models tend to encode patches, *e.g.*, `PatchTST` [32], or variates, *e.g.*, `iTransformer` [29]. This development inspires pre-training TSFMs on large-scale time serie datasets such as `TimesFM` [6], `Chronos` [1], `Moirai` [43], and `LightGTS` [41]. The adaptation of LLMs to time series include prompt-based methods, such as `PromptCast` [48] and `LLMTime` [10], and embedding-based methods, such as `GPT4TS` [55], `TimeLLM` [21], and `CALF` [27]. More recently, LVMs such as `MAE` [12] have been found more effective than LLMs by `VisionTS` [4], `VisionTS++` [36], and the study in [34, 51], which further inspires multimodal models for TSF such as `TimeVLM` [52] and `DMMV` [35]. In this work, we also include some CNN-based and MLP-based models as large models for their relatively complex designs, such as `SCINet` [26], `TimesNet` [44], and `TimeMixer` [40]. Despite their powerful performance, these large models are resource-demanding and may not fit resource-constrained scenarios.

Lightweight models for TSF. The SOTA lightweight TSF models mostly employ simple linear layers [23–25, 38, 47, 49, 50]. The rationale behind their effectiveness lies in the designs for exploring certain inductive biases pertaining to the TSF task. For example, DLinear [50] assumes linear mapping between lookback window and forecasts; CMoS [38] assumes linear correlation between historical chunks and future chunks; SparseTSF [25] and CycleNet [24] capitalize periodical patterns in TSF; while FITS [47] and FilterNet [49] take the advantage of frequency domain for efficient modeling of temporal trends. Despite the inspiring progress, none of these models can rival the SOTA performance of large models (Fig. 1) due to their non-large-model-aligned designs. In contrast, we are exploring the possibility of developing small models with large-model-like performance.

KD-based TSF. A straightforward way toward small models with large-model-like performance is KD. OccamVTS [30] is a recent cross-modal KD method that transfers TSF-essential knowledge from a pre-trained LVM teacher model to a smaller Transformer-based student model. However, as we evaluated in §4.4, OccamVTS's size still belongs to complex models. TimeDistill [31] supports cross-architecture KD, thus is more flexible in compressing student model size than OccamVTS. However, KD may not fit resource-constrained needs because (1) it relies on the availability (and fine-tuning) of large teacher models; (2) its training of student models involves communication with large teacher models, leading to high costs; and (3) the student model may need to be above certain size for sufficient capacity in encoding of teacher models' knowledge. As such, a standalone lightweight model may be more favorable.

3 The Proposed SVTime Model

Problem Statement. Given a multivariate time series (MTS) $\mathbf{X} = [\mathbf{x}^1, ..., \mathbf{x}^D]^\top \in \mathbb{R}^{D \times T}$ within a *look-back window* of length T, where $\mathbf{x}^i \in \mathbb{R}^T$ $(1 \leq i \leq D)$ is a univariate time series (UTS) of the i-th variate, the goal of LTSF is to estimate the most likely values of the MTS at future H time steps, *i.e.*, $\hat{\mathbf{Y}} \in \mathbb{R}^{D \times H}$, such that the difference between the estimation and the ground truth $\mathbf{Y} = \mathbf{X}_{T+1:T+H} \in \mathbb{R}^{D \times H}$ is minimized in terms of mean squared error (MSE), *i.e.*, $\frac{1}{D \cdot H} \sum_{i=1}^{D} \sum_{t=1}^{H} \|\hat{\mathbf{Y}}_{it} - \mathbf{Y}_{it}\|_2^2$ is minimized.

In the following, we introduce the key inductive biases (**IBs**) identified from LVM's forecasting behaviors, including (IB1) inter-period consistency (§3.1); (IB2) patch-wise variety (§3.2); and (IB3) distance-attenuating local attention (§3.3), meanwhile reprogramming them using linear layers and constraint functions, progressively constructing SVTime(-t) models. Finally, we encapsulate our models within a lightweight backcast-residual decomposition framework for avoiding overly dominant bias toward periods (§3.4). Fig. 5 illustrates the overall framework of SVTime(-t).

3.1 IB1: Inter-Period Consistency

Masked autoencoder (MAE) [12] is pre-trained self-supervisedly by reconstructing masked image patches using ImageNet dataset. To adapt it to LTSF, VisionTS [4] adopts a period-based imaging technique introduced by TimesNet [44]. Specifically, each length-T UTS $\mathbf{x}^i$ is segmented into $\lfloor T/P \rfloor$ subsequences of length P, where P is set to be the period of $\mathbf{x}^i$, which can be obtained using Fast Fourier Transform (FFT) on $\mathbf{x}^i$ [44] or from prior knowledge on
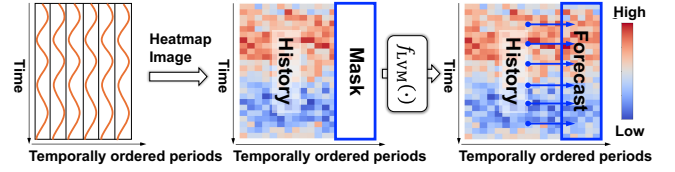

Figure 2: An illustration of inter-period consistency.

sampling frequency. The subsequences are stacked to form a 2D image $\mathbf{I}^i \in \mathbb{R}^{P \times \lfloor T/P \rfloor}$. After standard-deviation normalization, $\mathbf{I}^i$ is duplicated 3 times to form an image of size $P \times \lfloor T/P \rfloor \times 3$, followed by a bilinear interpolation to resize it to an image $\tilde{\mathbf{I}}^i$ of size $224 \times 224 \times 3$ to fit the input requirement of MAE.

As Fig. 2 shows, the forecast is achieved by reconstructing a right-appended masked area of $\tilde{\mathbf{I}}^i$, which corresponds to the future horizon of $\mathbf{x}^i$. The forecast $\hat{\mathbf{y}}^i \in \mathbb{R}^H$ can be recovered from the reconstructed area by de-normalization and reverse transformation. The forecast of MTS $\mathbf{X}$ is achieved by forecasting over $\mathbf{x}^1, ..., \mathbf{x}^D$ in parallel, following the channel-independence assumption [32].

Due to the period-based imaging and the spatial consistency enforced during MAE's pixel inference, VisionTS exhibits a strong bias toward **inter-period consistency** — smoothness of values over the same within-period time point (*i.e.*, rows) across periods (*i.e.*, columns) in the image, as revealed by [35, 51].

3.1.1 IB1-Informed Model Design. By designing a new small model, we don't need to perform bilinear interpolation and channel duplication. Instead, we use the 2D image $\mathbf{I}^i \in \mathbb{R}^{P \times \lfloor T/P \rfloor}$ as the input (*i.e.* the imaged look-back window) for the i-the variate, and output $\hat{\mathbf{I}}^i \in \mathbb{R}^{P \times \lfloor H/P \rfloor}$ as the forecasts for H time steps, where the j-th column $\hat{\mathbf{I}}_j^i \in \mathbb{R}^P$ $(1 \leq j \leq \lfloor H/P \rfloor)$ is the j-th forecasted period. $\hat{\mathbf{I}}^i$ corresponds to the masked area in Fig. 2.

To harness inter-period consistency, let $N = \lfloor T/P \rfloor$ and $M = \lfloor H/P \rfloor$. For each column $\hat{\mathbf{I}}_j^i$, we introduce a set of weights $\mathbf{w}_j = [w_{j,1}, w_{j,2}, ..., w_{j,N}]$ and predict $\hat{\mathbf{I}}_j^i$ as a linear combination of the historical periods in $\mathbf{I}^i$, *i.e.*, $\hat{\mathbf{I}}_j^i = \mathbf{I}^i \mathbf{w}_j^\top$. Let $\mathbf{W} = [\mathbf{w}_1^\top, ..., \mathbf{w}_M^\top] \in \mathbb{R}^{N \times M}$, the forecast of M periods is $\hat{\mathbf{I}}^i = \mathbf{I}^i \mathbf{W} \in \mathbb{R}^{P \times M}$. It is noteworthy that $\mathbf{W}$ is shared across variates. Then the forecasted time series $\hat{\mathbf{y}}^i \in \mathbb{R}^H$ can be recovered from $\hat{\mathbf{I}}^i$ in a similar way as aforementioned, and multiple variates for $1 \leq i \leq D$ in $\mathbf{X}$ can be forecasted in parallel according to the channel-independence assumption [32].
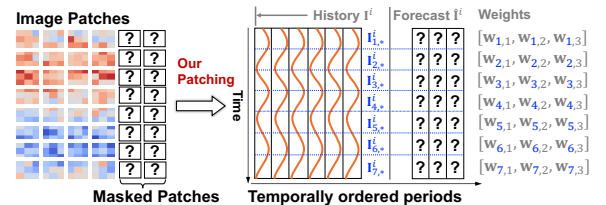
3.2 IB2: Patch-Wise Variety

Figure 3: An illustration of patch-wise variety.

As Fig. 3 illustrates, using ViT backbone [8], MAE divides an input image into a fixed number of patches, and encodes them for

reconstructing the patches in the masked area. This mechanism enables **patch-wise variety** — each row of patches in the image may have its row-specific inter-period consistency, while the degree of consistency may vary across different rows of patches.

3.2.1 IB2-Informed Model Design. To harness IB2, we divide each period in the historical image $\mathbf{I}^i$ into K patches, each is of length-$\lfloor P/K \rfloor$ (additional time points will be allocated to the last patch), as illustrated in Fig. 3 (where $K = 7$). Then the forecasting of $\hat{\mathbf{I}}^i$ will be performed in patch-wise.

Let the k-th row of the patches in $\mathbf{I}^i$ as $\mathbf{I}^i_{k,*} \in \mathbb{R}^{\lfloor P/K \rfloor \times N}$, and let the k-th patch in the j-th column of $\hat{\mathbf{I}}^i$ as $\hat{\mathbf{I}}^i_{k,j} \in \mathbb{R}^{\lfloor P/K \rfloor}$ ($j = 1, 2, 3$ in Fig. 3), we introduce weights $\mathbf{w}_{k,j} = [w_{k,j,1}, ..., w_{k,j,N}]$ and predict $\hat{\mathbf{I}}^i_{k,j}$ as a linear combination of the historical patches in $\mathbf{I}^i_{k,*}$, *i.e.*, $\hat{\mathbf{I}}^i_{k,j} = \mathbf{I}^i_{k,*} \mathbf{w}^\top_{k,j}$. Let $\mathbf{W}_k = [\mathbf{w}^\top_{k,1}, ..., \mathbf{w}^\top_{k,M}] \in \mathbb{R}^{N \times M}$, the forecast of the k-th patches for all of the M future periods becomes $\hat{\mathbf{I}}^i_{k,*} = \mathbf{I}^i_{k,*} \mathbf{W}_k \in \mathbb{R}^{\lfloor P/K \rfloor \times M}$. Forecasting $\hat{\mathbf{I}}^i_{k,*}$ for $1 \le k \le K$ in parallel accomplishes the forecasting of $\hat{\mathbf{I}}^i$, from which we can recover the forecasted time series $\hat{\mathbf{y}} \in \mathbb{R}^H$ as before.

Comparing with the model in §3.1.1, the model in this section extends a single $\mathbf{W}$ to $\mathbf{W}_1, ..., \mathbf{W}_K$ for fine-grained forecasting. As we evaluated in the ablation §4.3, this extension brings substantial performance improvement.

Remark. Our notion of "patch" refers specifically to *within-period patches*, and is used for inter-period consistency. It is different from the "patch" used in existing methods such as PatchTST [32], where a patch is a segment selected without using period.

3.3 IB3: Distance-Attenuating Local Attention

(a) LVM's Attention to historical patches for **(b)** Our Annealing Function
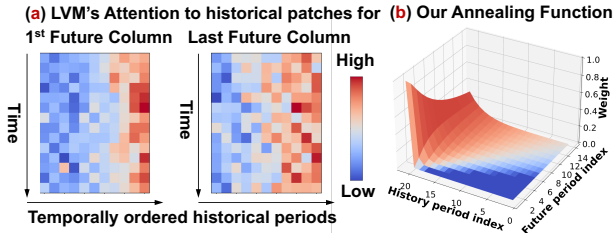1st Future Column   Last Future Column

Figure 4: An illustration of (a) distance-attenuating local attention; and (b) our annealing constraint function.

We also investigate the attention scores of MAE when forecasting different future periods (*i.e.*, different columns in the masked area of Fig 2). Fig. 4(a) shows the attention scores to the history for the first and last columns in the masked area. As can be seen, when forecasting the first period, the model focuses on the last several periods in the history, which are nearby to it. This *local attention*, however, *attenuates* when forecasting the last, *distant* period, where the model attention becomes more uniform across the entire history. This **distance-attenuating local attention** is reasonable since short-term forecasts may rely more on local patterns while long-term forecasts may depend on global patterns.

3.3.1 IB3-Informed Model Design. To encode this IB in our model, we propose to replace the weights $\mathbf{W}_1, ..., \mathbf{W}_K$ by a novel *annealing*

constraint function powered weighting mechanism. For the j-th period in the forecast $\hat{\mathbf{I}}^i$, the constraint function should generate weights $w_{j,1}, ..., w_{j,N}$ such that (1) when j is small (*e.g.*, close to 1), the weights focus on local areas, *i.e.*, weights close to $w_{j,N}$ are remarkably larger than those close to $w_{j,1}$; and (2) when j is large (*e.g.*, close to M), the weights tend to be uniform, *i.e.*, the differences of weights between $w_{j,1}$ and $w_{j,N}$ is small. To this end, we propose the following annealing function $w(j, n)$:

$$w(j, n) = \frac{\tilde{w}(j, n)}{\sum_{n=1}^N \tilde{w}(j, n)}, \text{ with } \tilde{w}(j, n) = \exp\left(\frac{\alpha \cdot (n - N)}{1 + \beta \cdot (j - 1)}\right) \quad (1)$$

where the numerator of $\tilde{w}(j, n)$ ensures $\tilde{w}(j, N)$ is larger than $\tilde{w}(j, 1)$ for a fixed j (*i.e.*, local attention). α is a parameter to scale the difference when n changes. The denominator of $\tilde{w}(j, n)$ smooths the weights when j is large. β is a parameter to control the degree of smoothness. $\exp(\cdot)$ ensures positive weights and the normalization in $w(j, n)$ makes the weights sum up to 1.

Fig. 4(b) shows the synergy between the numerator and denominator in Eq. (1), where short-term future periods have larger weights on nearby historical periods (N=20), while long-term future periods have more uniform weights, satisfying our design criteria.

Moreover, instead of manually tuning α and β as hyperparameters in Eq. (1), we introduce linear layers to learn each of them from the input time series $\mathbf{x}^i \in \mathbb{R}^T$ automatically:

$$\alpha^i = \text{SoftPlus}((\mathbf{w}^\alpha)^\top \mathbf{x}^i + b^\alpha), \ \beta^i = \text{SoftPlus}((\mathbf{w}^\beta)^\top \mathbf{x}^i + b^\beta) \quad (2)$$

where $\mathbf{w}^\alpha, \mathbf{w}^\beta \in \mathbb{R}^T$, b^α and b^β are parameters of the two linear layers. SoftPlus$(\cdot)$ is applied to avoid negative scaling. Eq. (2) enables different α^i's (β^i's) for different variates $\mathbf{x}^i$ ($1 \le i \le D$).

Finally, following IB2 in §3.2, we apply Eq. (1) to patches within periods. To this end, let $\tilde{\mathbf{W}} = [w(j, n)]^\top_{1 \le j \le M, 1 \le n \le N} \in \mathbb{R}^{N \times M}$ be the collection of weights from Eq. (1), we introduce learnable scalar weights $w^P_1, ..., w^P_K$ for K patches, and define $\tilde{\mathbf{W}}_k = w^P_k \tilde{\mathbf{W}}$ where $1 \le k \le K$. Then the set of weights $\tilde{\mathbf{W}}_1, ..., \tilde{\mathbf{W}}_K$, which encode our constraint function, replaces the fully learnable weights $\mathbf{W}_1, ... \mathbf{W}_K$ in §3.2.1, analogous to physics-informed learning. It is noteworthy that, this weighting mechanism only uses $O(T + K)$ parameters, in contrast to the $O(KMN)$ parameters in §3.2.1.

3.4 Backcast-Residual Decomposition

From IB1 (§3.1), LVMs tend to forecast periodic patterns while may overlook the global trend, as validated by [51]. To complement this bias, we encapsulate our model within a lightweight backcast-residual decomposition framework [35] as shown in Fig. 5(a).

Let LVM-IB$(\cdot)$ be the proposed block that encode IB1-IB3 (§3.1-§3.3). In addition to forecast $\hat{\mathbf{y}}^i$, the decomposition framework applies LVM-IB$(\cdot)$ to "backcast" the lookback window $\mathbf{x}^i$, *i.e.*, $[\hat{\mathbf{x}}^i, \hat{\mathbf{y}}^i] = $ LVM-IB$(\mathbf{x}^i)$ (the details of backcast is deferred to §3.5). Due to the period-prone prediction, $\hat{\mathbf{y}}^i$ tends to be forecasted periods, $\hat{\mathbf{x}}^i$ tend to be the seasonal component of $\mathbf{x}^i$, and the residual $\Delta \mathbf{x}^i = \mathbf{x}^i - \hat{\mathbf{x}}^i$ tend to be the trend component. Thus, $\Delta \mathbf{x}^i$ is fed to a linear layer to forecast the trend component $\Delta \hat{\mathbf{y}}^i \in \mathbb{R}^H$. Finally, $\Delta \hat{\mathbf{y}}^i$ is combined with the period forecast $\hat{\mathbf{y}}^i$ to determine the final forecast $\hat{\mathbf{y}}^i_{\text{final}}$. The overall process can be summarized as:

$$[\hat{\mathbf{x}}^i, \hat{\mathbf{y}}^i] = \text{LVM-IB}(\mathbf{x}^i) \rightarrow \Delta \mathbf{x}^i = \mathbf{x}^i - \hat{\mathbf{x}}^i \rightarrow$$
$$\Delta \mathbf{y}^i = \Delta \mathbf{x}^i \mathbf{W}^B + \mathbf{b}^B \rightarrow \hat{\mathbf{y}}^i_{\text{final}} = g \circ \Delta \mathbf{y}^i + (1 - g) \circ \hat{\mathbf{y}}^i \quad (3)$$
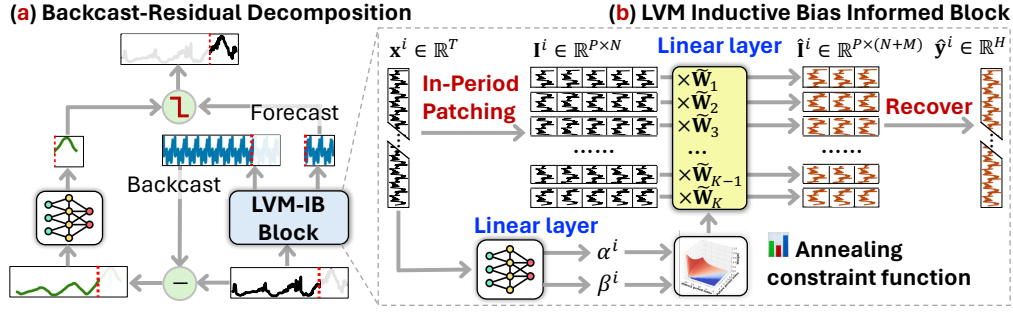
Figure 5: An overview of SVTime framework. (a) SVTime(-t) uses a backcast-residual decomposition to adaptively learn trend and seasonal components. (b) LVM-IB block uses linear layers and a constraint function to encode IB1-IB3 (§3.1-§3.3).

where $\mathbf{W}^B \in \mathbb{R}^{T \times H}$, $\mathbf{b}^B \in \mathbb{R}^H$ are parameters of the linear layer for trend forecasting, and $g = \text{sigmoid}(w^g) \in [0, 1]$ is a lightweight gate with a learnable scalar parameter w^g.

By doing so, the proposed SVTime(-t) can compensate the strong bias toward forecasting periods.

3.5 Summary of SVTime(-t) Model

Putting all components together, Fig. 5 summarizes the overall framework, where Fig. 5(b) shows the LVM-IB(·) block. In this paper, we study two instantiations of the LVM-IB(·) block, leading to two overall models: (1) **SVTime** uses only IB1 and IB2 and sets $\mathbf{W}_1$, ..., $\mathbf{W}_K$ as learnable parameters (*i.e.*, the model in §3.2.1); and (2) **SVTime-t** uses IB1-IB3 and configures $\tilde{\mathbf{W}}_1$, ..., $\tilde{\mathbf{W}}_K$ with annealing constraint function (*i.e.*, the model in §3.3.1).

They have different mechanisms for backcast in Eq. (3): (1) **SVTime** simply extends $\mathbf{W}_k$ from an N-by-M matrix to an N-by-$(N + M)$ matrix for all $1 \leq k \leq K$, *i.e.*, predicting N more periods in $\hat{\mathbf{x}}^i$ to represent an reconstruction of $\mathbf{x}^i$; (2) **SVTime-t** uses $\tilde{\mathbf{W}}_k$ ($1 \leq 1 \leq K$) to forecast $\hat{\mathbf{y}}^i$ while uses the learnable scalar $[w_1^p, ..., w_K^p]$ only (without constraint function) to backcast $\hat{\mathbf{x}}^i$.

Additionally, our LVM-IB(·) block allows stacking multiple blocks by properly setting input/output dimensions. For example,

$$[\hat{\mathbf{x}}_{l+2}^i, \hat{\mathbf{y}}^i] = \text{LVM-IB}(\hat{\mathbf{x}}_{l+1}^i), \text{ where } \hat{\mathbf{x}}_{l+1}^i = \text{LVM-IB}(\hat{\mathbf{x}}_l^i), \quad (4)$$

where $\hat{\mathbf{x}}_0^i = \mathbf{x}^i$. This allows re-parameterizing the history as $\hat{\mathbf{x}}_{l+1}^i$ before performing the final backcasting and forecasting. Empirically, this trick is found useful in improving forecasting performance.

Training. After obtaining $\hat{\mathbf{Y}} = [\hat{\mathbf{y}}^1, ..., \hat{\mathbf{y}}^D]$, SVTime(-t) is trained by minimizing MSE, *i.e.*, $\frac{1}{D \cdot H} \sum_{i=1}^D \sum_{t=1}^H \|\hat{\mathbf{Y}}_{it} - \mathbf{Y}_{it}\|_2^2$.

Complexity. The parameter size of a single LVM-IB(·) block in SVTime is $O(KN(M + N) + TH)$, and that of SVTime-t is $O(T + K + TH)$, which further reduces the size. Since K, M, N are small, the main overhead comes from TH — the linear layer in Eq. (3). However, as shown in Fig. 1, these sizes are competitively small, especially considering the earned performance gains.

4 Experiments

Datasets. We adopt 8 widely used MTS benchmarks: ETT (Electricity Transformer Temperature), including ETTh1, ETTh2, ETTm1,

ETTm2 [53]; Weather [53]; Electricity [53], Traffic [45]; and Solar-Energy [29]. Following standard protocols [32, 45], we split the datasets chronologically into training/validation/test sets using a 60%/20%/20% ratio for ETT and 70%/10%/20% for the others. The prediction horizon H is set to {96, 192, 336, 720} for all datasets. The look-back T is fixed at 512. The details about all of the datasets can be found in Appendix A.1.

The Compared Methods. We compare SVTime(-t) with 21 SOTA methods, including seven **lightweight models**: (1) DLinear [50]; (2) FITS [47], (3) TexFilter [49]; (4) PaiFilter [49]; (5) SparseTSF [25]; (6) CycleNet [24]; (7) CMoS [38]; eight **complex models**: (8) TimeMixer [40]; (9) TiDE [5]; (10) SCINet [26]; (11) TimesNet [44]; (12) iTransformer [29]; (13) PatchTST [32]; (14) FEDFormer [54]; (15) Autoformer [45]; six **pre-trained large models** covering **LVM**: (16) VisionTS [4]; **LLMs**: (17) GPT4TS [55]; (18) TimeLLM [21]; (19) CALF [27]; **VLM**: (20) TimeVLM [52]; and **TSFM**: (21) LightGTS [41]. Here, LightGTS is selected for its better performance than other SOTA TSFMs such as Moirai [28], Chronos [1], and Time-MoE [37], as reported in [41].

Additionally, we compare our method with a KD method — OccamVTS [30] — specifically in §4.5. For our method, we evaluate both SVTime and SVTime-t. The hyperparameter K — the number of patches (§3.2.1) — is set as $\lfloor P/6 \rfloor$. We analyze the impact of K in §4.5. The number of LVM-IB(·) block is searched within $[1, 3]$ using validation set. Ablation studies include several variants of our methods (§4.3). Following [4], the imaging period P for VisionTS, SVTime, and SVTime-t is set based on each dataset's sampling frequency (see Appendix A.1).

Evaluation. Following [32, 39, 50], we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) to evaluate the LTSF performance of the compared methods, and use parameter size, GPU cost, training/inference time, *etc.*, to evaluate model complexity.

4.1 Experiment Results

Table 1 presents the LTSF performance of the 7 lightweight models, most of which are composed of linear layers. Table 2 summarizes the LTSF performance of the 8 complex models, including MLP-based, CNN-based, and Transformer-based architectures. All models were trained 3 times with NVIDIA RTX 6000 Ada GPUs. The averaged

Table 1: LTSF performance comparison of lightweight models on benchmark datasets. The results are averaged over 5 runs across prediction horizons $H \in \{96, 192, 336, 720\}$. Lower MSE and MAE indicate better performance. Red (blue) values indicate the best (second-best) MSE and MAE per row. Full results are available in Appendix B.2.

Model	ETTh1 MSE	ETTh1 MAE	ETTh2 MSE	ETTh2 MAE	ETTm1 MSE	ETTm1 MAE	ETTm2 MSE	ETTm2 MAE	Weather MSE	Weather MAE	Electricity MSE	Electricity MAE	Traffic MSE	Traffic MAE	Solar MSE	Solar MAE	Wins
DLinear (2022)	0.447	0.459	0.442	0.447	0.370	0.396	0.264	0.329	0.245	0.300	0.163	0.261	0.421	0.294	0.232	0.297	0 (0)
FITS (2023)	0.430	0.439	0.346	0.392	0.369	0.387	0.258	0.317	0.245	0.283	0.296	0.400	0.431	0.305	0.774	0.705	1 (2)
TexFilter (2024)	0.424	0.446	0.372	0.410	0.371	0.399	0.288	0.340	0.232	0.270	0.167	0.264	0.415	0.299	0.209	0.271	0 (1)
PaiFilter (2024)	0.429	0.442	0.371	0.409	0.364	0.390	0.265	0.326	0.223	0.262	0.165	0.259	0.416	0.294	0.199	0.255	3 (1)
SparseTSF (2024)	0.425	0.444	0.360	0.399	0.363	0.382	0.256	0.314	0.244	0.281	0.197	0.292	0.432	0.298	0.237	0.269	2 (0)
CycleNet (2024)	0.419	0.431	0.358	0.396	0.364	0.386	0.258	0.317	0.241	0.279	0.158	0.251	0.412	0.287	0.229	0.289	0 (3)
CMoS (2025)	0.416	0.431	0.355	0.399	0.363	0.384	0.267	0.323	0.231	0.269	0.165	0.257	0.424	0.286	0.224	0.263	1 (2)
SVTime-t (Ours)	0.417	0.430	0.357	0.399	0.358	0.379	0.259	0.316	0.231	0.269	0.165	0.258	0.419	0.286	0.232	0.273	0 (6)
SVTime (Ours)	0.418	0.421	0.351	0.386	0.346	0.369	0.265	0.321	0.240	0.280	0.157	0.247	0.378	0.248	0.213	0.245	9 (1)

Table 2: LTSF perforamnce comparison of complex models on benchmark datasets. Full results are available in Appendix B.2.

	Model	ETTh1 MSE	ETTh1 MAE	ETTh2 MSE	ETTh2 MAE	ETTm1 MSE	ETTm1 MAE	ETTm2 MSE	ETTm2 MAE	Weather MSE	Weather MAE	Electricity MSE	Electricity MAE	Traffic MSE	Traffic MAE	Solar MSE	Solar MAE	Wins
MLP	TimeMixer (2024)	0.470	0.475	0.353	0.402	0.429	0.428	0.312	0.354	0.244	0.280	0.185	0.286	0.438	0.320	0.225	0.280	0 (0)
MLP	TiDE (2023)	0.421	0.433	0.343	0.389	0.366	0.385	0.257	0.315	0.243	0.280	0.165	0.258	0.436	0.313	0.236	0.275	3 (2)
CNN	SCINet (2021)	0.483	0.472	0.399	0.428	0.427	0.432	0.296	0.347	0.259	0.294	0.217	0.324	0.510	0.400	0.227	0.310	0 (0)
CNN	TimesNet (2022)	0.538	0.514	0.397	0.434	0.446	0.438	0.323	0.358	0.275	0.305	0.214	0.311	0.623	0.335	0.216	0.287	0 (0)
Transformer	Autoformer (2021)	0.544	0.535	0.438	0.480	0.569	0.505	0.340	0.389	0.355	0.397	0.276	0.375	0.666	0.407	0.848	0.692	0 (0)
Transformer	FEDFormer (2022)	0.480	0.498	0.437	0.480	0.432	0.458	0.343	0.392	0.366	0.413	0.231	0.343	0.610	0.373	0.330	0.415	0 (0)
Transformer	PatchTST (2023)	0.434	0.452	0.347	0.390	0.354	0.385	0.259	0.321	0.226	0.265	0.164	0.258	0.396	0.271	0.187	0.250	3 (6)
Transformer	iTransformer (2023)	0.465	0.470	0.396	0.422	0.372	0.401	0.272	0.332	0.235	0.274	0.161	0.258	0.398	0.284	0.206	0.269	0 (2)
	SVTime-t (Ours)	0.417	0.430	0.357	0.399	0.358	0.379	0.259	0.316	0.231	0.269	0.165	0.258	0.419	0.286	0.232	0.273	1 (6)
	SVTime (Ours)	0.418	0.421	0.351	0.386	0.346	0.369	0.265	0.321	0.240	0.280	0.157	0.247	0.378	0.248	0.213	0.245	9 (1)

MSE and MAE across all prediction horizons $H \in \{96, 192, 336, 720\}$ are reported. The full results can be found in Appendix B.2.

From Table 1, several key insights emerge: (1) From Fig. 1, the smallest models are CMoS, FITS, and SparseTSF, which usually encode a single hypothesis, *e.g.*, correlation among historical chunks in CMoS, simplifying their designs. The simplification may underfit some complex datasets, leading to their inferior performance in Table 1; (2) The best baseline appears to be PaiFilter, whose parameter size is at a similar level as our SVTime, while being larger than SVTime-t, indicating a less effective use of parameters; (3) Our SVTime shows consistent superiority over the lightweight baselines, achieving 9 first-places (39 first-places in Table 7 of Appendix B.2), confirming its non-trivial design inspired by LVM biases, and suggesting a best trade-off between performance and model size; and (4) Our SVTime-t further reduces the model size, thus is less powerful, but still achieves 6 second-places, surpassing the baselines in most cases.

Moreover, from Table 2, complex models — whose parameter sizes range from 400K to 10.5M (Fig. 1(c), *i.e.*, ~ 2× to 50× larger than SVTime — do not show superiority over lightweight models. This observation is consistent with some existing studies [24, 25]. The best complex baseline appears to be PatchTST, confirming its well-accepted design of time series patching (note: different from our patching, as discussed in §3.2.1) and channel-independence assumption. Whereas, it needs 3M+ parameters. In contrast, SVTime(-t) outperform these baselines in most cases, particularly in handling datasets that contain anomalies such as ETTm1 (see §4.5).

These results underscore SVTime's potential as a powerful small model, while suggest SVTime-t as a less powerful yet more compact model for practical deployment in resource-constrained scenarios.

4.2 Comparing with Pretrained Large Models

Fig. 6(a)(b) compare SVTime(-t) with the six pre-trained large models on Electricity and Traffic datasets after fine-tuning with the training sets. The full results on other datasets can be found in Appendix B.4. From the figures, we observe (1) Significant difference in parameter size and inference time between SVTime(-t) and other large models. For example, SVTime only uses 0.2% parameters of VisionTS, and 0.003% parameters of TimeLLM; (2) VisionTS performs better than other large models, confirming the rationale of using LVM to inspire our small models; (3) Competitive performance of SVTime, which shows 3% (2%) improvement over VisionTS on Electricity (Traffic) dataset, and may be the only small model in our experiments that can rival large models, without any pre-training; and (4) SVTime-t, despite its slightly lower performance than VisionTS, still outperforming some large models such as GPT4TS, CALF, and LightGTS, suggesting its potential in energy-saving scenarios.

4.3 Ablation Analysis

We validate the design of SVTime(-t) through ablation studies on all datasets. Table 3 summarizes the analysis: (a) removes IB2 from

Table 3: Ablation analysis of SVTIME(-t). MSE and MAE are averaged over different prediction lengths. Lower MSE and MAE are better. "Improvement"s of ablations (a)(b) (or (c)(d)(e)) are relative to SVTIME (or SVTIME-t).

Method	ETTh1		ETTh2		ETTm1		ETTm2		Weather		Electricity		Traffic		Solar	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
SVTIME	0.418	**0.421**	0.351	**0.386**	**0.346**	**0.369**	0.265	0.321	**0.240**	0.280	**0.157**	**0.247**	**0.378**	**0.248**	**0.213**	**0.245**
(a) - IB2	0.420	0.429	**0.342**	0.389	0.361	0.380	**0.254**	**0.314**	0.243	**0.279**	0.165	0.257	0.485	0.344	0.233	0.264
Improvement	-0.48%	-1.90%	2.56%	-0.78%	-4.34%	-2.98%	4.15%	2.18%	-1.25%	0.36%	-5.10%	-4.05%	-28.31%	-38.71%	-9.39%	-7.35%
(b) - Backcast	**0.411**	**0.421**	0.369	0.402	0.420	0.428	0.287	0.340	0.263	0.301	0.214	0.295	0.518	0.282	0.418	0.434
Improvement	1.67%	0.00%	-5.13%	-4.15%	-21.39%	-15.99%	-8.30%	-5.92%	-9.58%	-7.50%	-36.31%	-19.43%	-37.04%	-13.71%	-96.24%	-77.14%
SVTIME-t	**0.417**	0.430	**0.357**	0.399	**0.358**	0.379	**0.259**	0.316	**0.231**	0.269	**0.165**	0.258	**0.419**	0.286	**0.232**	0.273
(c) -IB2	0.422	**0.428**	0.366	0.411	0.361	**0.377**	0.289	0.336	0.293	0.315	0.170	0.261	0.425	0.284	0.251	**0.272**
Improvement	-1.20%	0.47%	-2.52%	-3.01%	-0.84%	0.53%	-11.58%	-6.33%	-26.84%	-17.10%	-3.03%	-1.16%	-1.43%	0.70%	-8.19%	0.37%
(d) -IB3	0.428	0.440	0.359	**0.399**	0.362	0.378	0.289	0.336	0.243	0.280	0.169	0.260	0.425	**0.283**	0.251	0.273
Improvement	-2.64%	-2.33%	-0.56%	0.00%	-1.12%	0.26%	-11.58%	-6.33%	-5.19%	-4.09%	-2.42%	-0.78%	-1.43%	1.05%	-8.19%	0.00%
(e) - Backcast	0.488	0.510	0.371	0.404	0.565	0.449	0.313	0.354	0.295	0.319	0.212	0.293	0.622	0.387	0.385	0.423
Improvement	-17.03%	-18.60%	-3.92%	-1.25%	-57.82%	-18.47%	-20.85%	-12.03%	-27.71%	-18.59%	-28.48%	-13.57%	-48.45%	-35.31%	-65.95%	-54.95%

(a) Comparison on Electricity dataset
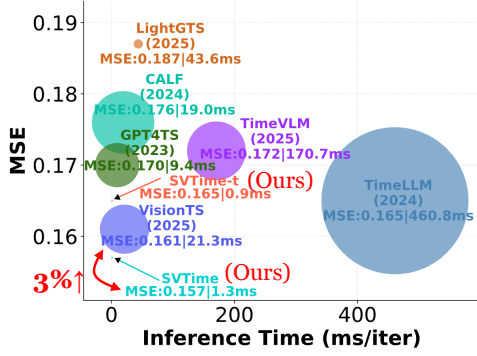
(b) Comparison on Traffic dataset
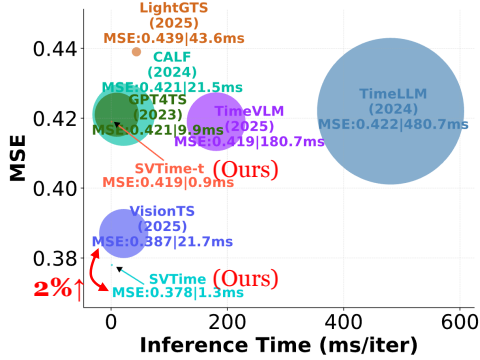
Figure 6: LTSF performance comparison with pre-trained large models on (a) Electricity and (b) Traffic datasets. Bubble size is proportional to the parameter size of each model.

SVTIME, *i.e.*, without using $\mathbf{W}_1, ..., \mathbf{W}_K$ for enabling patch-wise variety. This is equivalent to the model with only IB1 (§3.1.1), *i.e.*, using a single $\mathbf{W}$; (b) removes the backcast-residual decomposition framework (§3.4) from SVTIME; (c) removes IB2 from SVTIME-t, which is equivalent to set $K = 1$ for $\tilde{\mathbf{W}}_1, ..., \tilde{\mathbf{W}}_K$ (see §3.3.1), *i.e.*, disabling patch-wise variety; (d) removes IB3 from SVTIME-t, *e.g.*, removing the annealing constraint function by simply setting $w(j, n) = 1$

Table 4: Comparison between SVTIME(-t) and OccamVTS.

Metrics	SVTIME	SVTIME-t	OccamVTS
MSE	**0.346**	0.358	<u>0.349</u>
MAE	**0.369**	0.379	<u>0.372</u>
GPU memory (MiB)	1,095	692	11,859
Train time (s/epoch)	2.9	2.6	275
Inference time (ms/iter)	1.08	0.84	2.83
Parameter size	215.5K	162.7K	2,834.4K

in Eq. (1) for $\forall 1 \leq j \leq M, 1 \leq n \leq N$, which is equivalent to set $\tilde{\mathbf{W}}_k = w_k^p$ for $1 \leq k \leq K$; and (e) removes the backcast-residual decomposition framework from SVTIME-t. Note that IB1 cannot be removed as it establishes the basis of the proposed models.

Table 3 reveals several key insights into the design of SVTIME(-t). In (a)(c), removing IB2 — patch-wise variety — from the modeling of historical periods degrades the performance of SVTIME(-t) in most cases, confirming the effectiveness of fine-grained modeling of within-period patches using $\mathbf{W}_k$ (or $\tilde{\mathbf{W}}_k$) ($1 \leq k \leq K$). In (b)(e), we observe a major performance drop by removing the backcast-residual decomposition, highlighting the disadvantage of solely modeling periodical patterns while suggesting the effectiveness of the adaptive decomposition in mitigating this bias. Finally, (d) underscores the importance of the proposed annealing constraint function: removing it leads to uniform attention to historical periods, which contradicts LVM's distance-attenuating local attention, indirectly validating IB3's usefulness in LVM forecasters.

Overall, the patch-wise variety, distance-attenuating local attention, and the backcast-residual decomposition are crucial to the success of SVTIME and SVTIME-t.

4.4 Comparing with Knowledge Distillation

We compare SVTIME(-t) with OccamVTS [30] — a knowledge distillation (KD) based student model trained using MAE as its teacher model — using EETm1 dataset as an example (results on other datasets are similar thus are omitted for brevity).

Table 4 summarizes the comparison in terms of both LTSF performance and computational costs. All models were trained with the same batch size of 512. The results are averaged over all prediction horizons $H \in \{96, 192, 336, 720\}$. From Table 4, we observe (1) all of
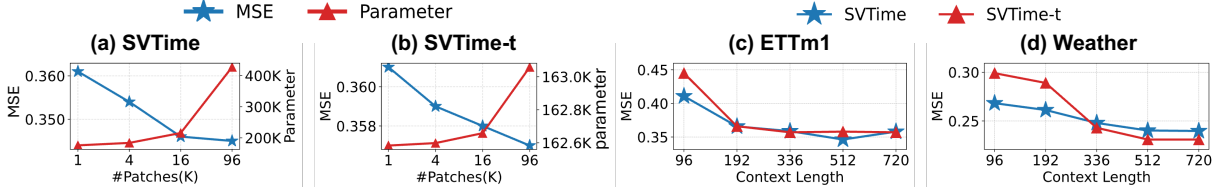
Figure 7: Results on the impact of the number of patches K on (a) SVTime; and (b) SVTime-t in terms of LTSF performance and model size; and the impact of lookback window (or context) length on both models using (c) ETTm1; and (d) Weather datasets.
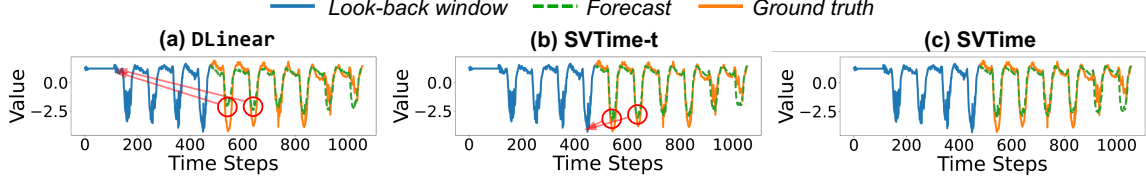

Figure 8: Comparison of (a) `DLinear`; (b) SVTime-t; and (c) SVTime on the same example in the ETTm1 dataset that highlights the importance of distance-attenuating local attention.

the three models demonstrate comparable performance, while SV-Time slightly outperforms `OccamVTS`, suggesting all of the models inherit the merits of the LVM forecaster, either through "physics" guided learning or KD; (2) `OccamVTS` may not be considered as a lightweight model for its 2.8M parameter size. This is because it uses Transformer backbone — the same as its teacher model — and doesn't support cross-architecture KD; (3) SVTime-t is the most efficient model with 17× less memory, 106× faster training, and 3× faster inference than `OccamVTS`. Also, SVTime's cost is more appealing than `OccamVTS`. The high training cost of `OccamVTS` is due to its involvement of a large teacher model `MAE`; and (4) we note that `OccamVTS`'s reliance on a large teacher model needs extra costs for training and fine-tuning the teacher model before KD. As such, SVTime(-t) offers advantages over KD-based models.

4.5 Performance Analysis

In this section, we perform an in-depth analysis of SVTime(-t).

Impact of the number of patches. Fig.7(a)(b) assess the impact of the number of within-period patches, *i.e.*, K (§3.2.1) on the performance of SVTime and SVTime-t when fixing period length $P = 96$ on the ETTm1 dataset. Also, since increasing K leads to more parameters in $\mathbf{W}_1, ..., \mathbf{W}_K$ (for SVTime) and $\tilde{\mathbf{W}}_1, ..., \tilde{\mathbf{W}}_K$ (for SVTime-t), we evaluate their parameter sizes. From Fig.7(a)(b), both models achieve better performance with a larger K, which corresponds to more fine-grained patching. Notably, for SVTime, the performance improvement from $K = 16$ to $K = 96$ is small while the model size expands a lot, suggesting a saturating point at $K = 16$. For SVTime-t, the changing ranges of MSE and model size caused by varying K are relatively small. This is because $\tilde{\mathbf{W}}_1, ..., \tilde{\mathbf{W}}_K$ only have $O(K)$ learnable parameters, thus varying K has a small impact. Overall, for both models, we set $K = \lfloor P/6 \rfloor$ for the best trade-off between performance and efficiency.

Distance-attenuating local attention. Fig. 8 compares SVTime SVTime-t, and `DLinear` on a sample ETTm1 case that highlights

the importance of distance-attenuating local attention (§3.3). The lookback window starts with a flattened high value and ends with a notable dip. `DLinear` uses a linear layer to forecast without any constraint, thus doesn't guarantee local attention. In fact, Fig. 8(a) shows its first two predicted dips might be influenced by the high value in the lookback window, while the first two ground truth dips follow the nearby dip in the lookback window, leading to errors. In contrast, SVTime-t effectively uses its constraint function to predict the first two dips. SVTime slightly underperforms but is still better than `DLinear`. Moreover, SVTime-t's forecast of all dips gradually rises due to its attenuating local attention, which matches the ground truth. However, `DLinear` has no such pattern.

Impact of lookback window. Fig. 7(c)(d) assess SVTime(-t)'s performance *w.r.t.* varying lookback window length on ETTm1 and Weather datasets. Using MSE metric, we observe both SVTime and SVTime-t benefit from longer lookback windows. SVTime is more effective than SVTime-t when the lookback window is small. This is possibly because a shorter lookback window may lead to more constrained local weights by SVTime-t's annealing constraint function, resulting in a slight sensitivity to a small context length.

5 Conclusion

This paper introduces SVTime and SVTime-t, novel lightweight models whose design is informed by the inductive biases — analogous to the "physics" governing the behaviors — of a powerful LVM forecaster. Through the discovery and re-programming of three inductive biases, *i.e.*, inter-period consistency, patch-wise variety, and distance-attenuating local attention, along with the adoption of a backcast-residual decomposition framework, SVTime(-t) effectively inherits the merits of the LVM forecaster, while mitigating the dominant bias of period-prone forecasting. Extensive experiments on 8 benchmark datasets with 21 SOTA baselines covering lightweight, complex, pre-trained large models, and a KD model demonstrate SVTime(-t)'s large-model-like performance and economical size, highlighting the potential in energy-saving scenarios.

References

[1] Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, et al. 2024. Chronos: Learning the Language of Time Series. *Trans. Mach. Learn. Res.* (2024).

[2] Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small Language Models are the Future of Agentic AI. *arXiv* (2025).

[3] Verónica Bolón-Canedo, Laura Morán-Fernández, Brais Cancela, and Amparo Alonso-Betanzos. 2024. A review of green artificial intelligence: Towards a more sustainable future. *Neurocomputing* (2024).

[4] Mouxiang Chen, Lefei Shen, Zhuo Li, Xiaoyun Joy Wang, Jianling Sun, and Chenghao Liu. 2025. VisionTS: Visual Masked Autoencoders Are Free-Lunch Zero-Shot Time Series Forecasters. In *ICLR*.

[5] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan K Mathur, Rajat Sen, and Rose Yu. 2023. Long-term Forecasting with TiDE: Time-series Dense Encoder. *Trans. Mach. Learn. Res.* (2023).

[6] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. 2024. A decoder-only foundation model for time-series forecasting. In *ICML*.

[7] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, et al. 2024. Hymba: A hybrid-head architecture for small language models. *arXiv* (2024).

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

[9] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. 2024. MOMENT: A Family of Open Time-series Foundation Models. In *ICML*.

[10] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. 2023. Large language models are zero-shot time series forecasters. *NeurIPS* (2023).

[11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv* (2025).

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *CVPR*.

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked Autoencoders Are Scalable Vision Learners. In *CVPR*.

[14] Kethmi Hirushini Hettige, Jiahao Ji, Shili Xiang, Cheng Long, Gao Cong, and Jingyuan Wang. 2024. AirPhyNet: Harnessing Physics-Guided Neural Networks for Air Quality Prediction. In *ICLR*.

[15] Min Hou, Chang Xu, Zhi Li, Yang Liu, Weiqing Liu, Enhong Chen, and Jiang Bian. 2022. Multi-granularity residual learning with confidence estimation for time series prediction. In *WWW*.

[16] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog* (2023).

[17] Sheo Yon Jhin, Jaehoon Lee, Minju Jo, Seungji Kook, Jinsung Jeon, Jihyeon Hyeong, Jayoung Kim, and Noseong Park. 2022. Exit: Extrapolation and interpolation-based neural controlled differential equations for time-series classification and forecasting. In *WWW*.

[18] Jiahao Ji, Jingyuan Wang, Zhe Jiang, Jiawei Jiang, and Hu Zhang. 2022. STDEN: Towards physics-guided neural networks for traffic flow prediction. In *AAAI*.

[19] Renhe Jiang, Zhaonan Wang, Yudong Tao, Chuang Yang, Xuan Song, Ryosuke Shibasaki, Shu-Ching Chen, and Mei-Ling Shyu. 2023. Learning social meta-knowledge for nowcasting human mobility in disaster. In *WWW*.

[20] Xinrui Jiang, Yicheng Pan, Meng Ma, and Ping Wang. 2023. Look deep into the microservice system anomaly through very sparse logs. In *WWW*.

[21] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. 2024. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. In *ICLR*.

[22] Harshavardhan Kamarthi, Lingkai Kong, Alexander Rodríguez, Chao Zhang, and B Aditya Prakash. 2022. CAMul: calibrated and accurate multi-view time-series forecasting. In *WWW*.

[23] Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. 2023. Revisiting Long-term Time Series Forecasting: An Investigation on Linear Mapping. arXiv:2305.10721 [cs.LG]

[24] Shengsheng Lin, Weiwei Lin, Xinyi Hu, Wentai Wu, Ruichao Mo, and Haocheng Zhong. 2024. Cyclenet: Enhancing time series forecasting through modeling periodic patterns. In *NeurIPS*.

[25] Shengsheng Lin, Weiwei Lin, Wentai Wu, Haojun Chen, and Junjie Yang. 2024. SparseTSF: Modeling Long-term Time Series Forecasting with* 1k* Parameters. In *ICML*.

[26] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. 2022. Scinet: Time series modeling and forecasting with sample convolution and interaction. In *NeurIPS*.

[27] Peiyuan Liu, Hang Guo, Tao Dai, Naiqi Li, Jigang Bao, Xudong Ren, Yong Jiang, and Shu-Tao Xia. 2025. Calf: Aligning llms for time series forecasting via cross-modal fine-tuning. In *AAAI*.

[28] Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Junnan Li, Silvio Savarese, Caiming Xiong, et al. 2025. Moirai-MoE: Empowering Time Series Foundation Models with Sparse Mixture of Experts. In *ICML*.

[29] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2023. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *ICLR*.

[30] Sisuo Lyu, Siru Zhong, Weilin Ruan, Qingxiang Liu, Qingsong Wen, Hui Xiong, and Yuxuan Liang. 2025. OccamVTS: Distilling Vision Models to 1% Parameters for Time Series Forecasting. *arXiv* (2025).

[31] Juntong Ni, Zewen Liu, Shiyu Wang, Ming Jin, and Wei Jin. 2025. Timedistill: Efficient long-term time series forecasting with mlp via cross-architecture distillation. *arXiv* (2025).

[32] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *ICLR*.

[33] Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. 2024. S2IP-LLM: Semantic Space Informed Prompt Learning with LLM for Time Series Forecasting. In *ICML*.

[34] Weilin Ruan, Siru Zhong, Haomin Wen, and Yuxuan Liang. 2025. Vision-Enhanced Time Series Forecasting via Latent Diffusion Models. *arXiv* (2025).

[35] ChengAo Shen, Wenchao Yu, Ziming Zhao, Dongjin Song, Wei Cheng, Haifeng Chen, and Jingchao Ni. 2025. Multi-Modal View Enhanced Large Vision Models for Long-Term Time Series Forecasting. In *NeurIPS*.

[36] Lefei Shen, Mouxiang Chen, Xu Liu, Han Fu, Xiaoxue Ren, Jianling Sun, Zhuo Li, and Chenghao Liu. 2025. VisionTS++: Cross-Modal Time Series Foundation Model with Continual Pre-trained Visual Backbones. *arXiv* (2025).

[37] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. [n. d.]. Time-MoE: Billion-Scale Time Series Foundation Models with Mixture of Experts. In *ICLR*.

[38] Haotian Si, Changhua Pei, et al. 2025. CMoS: Rethinking Time Series Prediction Through the Lens of Chunk-wise Spatial Correlations. In *ICML*.

[39] Mingtian Tan, Mike Merrill, Vinayak Gupta, Tim Althoff, and Tom Hartvigsen. 2024. Are language models actually useful for time series forecasting?. In *NeurIPS*.

[40] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and JUN ZHOU. 2024. TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting. In *ICLR*.

[41] Yihang Wang, Yuying Qiu, Peng Chen, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. 2025. LightGTS: A Lightweight General Time Series Forecasting Model. In *ICML*.

[42] Wei Wei, Chao Huang, Lianghao Xia, and Chuxu Zhang. 2023. Multi-modal self-supervised learning for recommendation. In *WWW*.

[43] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2024. Unified Training of Universal Time Series Forecasting Transformers. In *ICML*.

[44] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *ICLR*.

[45] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*.

[46] Wentao Xu, Weiqing Liu, Chang Xu, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2021. Rest: Relational event-driven stock trend forecasting. In *WWW*.

[47] Zhijian Xu, Ailing Zeng, and Qiang Xu. 2024. FITS: Modeling Time Series with 10k Parameters. In *ICLR*.

[48] Hao Xue and Flora D Salim. 2023. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Trans. Knowl. Data Eng.* (2023).

[49] Kun Yi, Jingru Fei, Qi Zhang, Hui He, Shufeng Hao, Defu Lian, and Wei Fan. 2024. Filternet: Harnessing frequency filters for time series forecasting. In *NeurIPS*.

[50] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *AAAI*.

[51] Ziming Zhao, ChengAo Shen, Hanghang Tong, Dongjin Song, Zhigang Deng, Qingsong Wen, and Jingchao Ni. 2025. From Images to Signals: Are Large Vision Models Useful for Time Series Analysis? *arXiv* (2025).

[52] Siru Zhong, Weilin Ruan, Ming Jin, Huan Li, Qingsong Wen, and Yuxuan Liang. 2025. Time-VLM: Exploring Multimodal Vision-Language Models for Augmented Time Series Forecasting. In *ICML*.

[53] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hao Xiong, Wancai Zhang, and Weinan Xie. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*.

[54] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*. PMLR, 27268–27286.

[55] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. 2023. One fits all: Power general time series analysis by pretrained lm. *NeurIPS* (2023).

A   Experiment Setting

A.1   Datasets

Following [26, 29, 38, 44], our experiments are conducted on 8 widely used LTSF benchmark datasets that cover a wide range of sampling frequencies, number of variates, levels of periodicity, and real-world domains. The four ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2) record oil temperature from two electric transformers, sampled at 15-minute and hourly intervals. The Weather dataset collects measurements of meteorological indicators in Germany every 10 minutes. The Electricity dataset records hourly electricity consumption of Portuguese clients. The Traffic dataset measures hourly road occupancy rates from sensors on San Francisco freeways. The Solar contains hourly solar power output measurements from U.S. photovoltaic plants. Table 5 summarizes the statistics of the datasets.

Table 5: Statistics of the benchmark datasets. "Dataset Size" is organized in (Train, Validation, Test).

Dataset	# Variates	Series Length	Dataset Size	Frequency
ETTh1	7	17420	(8545, 2881, 2881)	Hourly
ETTh2	7	17420	(8545, 2881, 2881)	Hourly
ETTm1	7	69680	(34465, 11521, 11521)	15 mins
ETTm2	7	69680	(34465, 11521, 11521)	15 mins
Weather	321	52696	(36792, 5271, 10540)	10 mins
Electricity	21	26304	(18317, 2633, 5261)	Hourly
Traffic	862	17544	(12185, 1757, 3509)	Hourly
Solar	137	52560	(36792,5768,11024)	Hourly

A.2   Running Environment

The experiments are conducted on a Linux server (kernel 5.15.0-139) with NVIDIA RTX 6000 Ada GPUs (48 GB). The environment uses Python 3.12.8, PyTorch 2.5.1 with CUDA 12.4 and cuDNN 9.1. The key libraries include NumPy 2.1.3, Pandas 2.2.3, Matplotlib 3.10.0, SciPy 1.15.1, scikit-learn 1.6.1, and torchvision 0.20.1.

B   More Experimental Results

B.1   Statistical Stability

To ensure the statistical reliability of our results, we trained and evaluated all models three times on each benchmark dataset using different random seeds (*seed* = 2021 ∼ 2023). Table 6 reports the average standard deviations of SVTime (-t) across the eight datasets. As shown in the table, the maximum standard deviation is below 0.004, indicating that SVTime(-t) exhibits strong stability and robustness.

B.2   Full Results with baseline Models

Tables 7 and Table 8 provide the complete comparison of SVTime (-t) with seven light-weight models and eight complex models from different categories, respectively, serving as a supplement to Table 1 and Table 2 in the main paper.

From these tables, we can observe that SVTime maintains a clear advantage when compared against all baseline methods. Against the light-weight models, it achieves 39 first-place results; against the complex models, it achieves 35 first-place results. Although

Table 6: Standard deviations of SVTime (-t) across all baselines. Results computed over three independent runs with random seeds (*seed* = 2021 ∼ 2023).

Dataset	SVTime-t		SVTime	
	MSE	MAE	MSE	MAE
ETTh1	0.417 ± 0.001	0.430 ± 0.001	0.418 ± 0.001	0.421 ± 0.001
ETTh2	0.358 ± 0.003	0.399 ± 0.001	0.351 ± 0.003	0.386 ± 0.001
ETTm1	0.358 ± 0.002	0.379 ± 0.001	0.346 ± 0.002	0.369 ± 0.001
ETTm2	0.260 ± 0.001	0.316 ± 0.001	0.265 ± 0.004	0.321 ± 0.002
Weather	0.231 ± 0.001	0.269 ± 0.001	0.240 ± 0.002	0.280 ± 0.003
Electricity	0.165 ± 0.000	0.258 ± 0.000	0.157 ± 0.003	0.247 ± 0.003
Traffic	0.419 ± 0.000	0.287 ± 0.001	0.378 ± 0.001	0.248 ± 0.003
Solar	0.232 ± 0.001	0.274 ± 0.001	0.213 ± 0.001	0.245 ± 0.002

SVTime-t shows some performance degradation, it still obtains 19 second-best or better results when compared with the light-weight models and 26 second-best or better results when compared with the complex models. These results demonstrate that our SVTime (-t) as a light-weight model, remains highly competitive with both light-weight and complex baselines. And these results further validate that incorporating inductive biases from vision models is an effective strategy for designing small yet powerful models.

B.3   Full Ablation Results

In Table 3, we present the ablation analysis of SVTime (-t), where the MSE and MAE values are averaged over different prediction horizons. Table 9 further provides the complete results covering all prediction lengths $H \in \{96, 192, 336, 720\}$ for comprehensive reference. We observe that removing the patch-wise variety, the distance-attenuating local attention, or the back-cast mechanism at different horizons leads to a clear performance degradation. This further demonstrates the indispensable role of these components in SVTime (-t), as they collectively contribute significantly to the overall forecasting performance.

B.4   Comparison with Pre-trained Models

Table 10, as a complement to Fig. 6, provides a more extensive comparison between SVTime and various large pre-trained models on general time series datasets. CALF[27], GPT4TS[55] and TimeLLM[21] are based on language pretrained models; VisionTS[4] leverages vision pretrained models; TimeVLM[52] uses vision and language pretrained models; and LightGTS[41] is pretrained purely on time series data. Our model achieves the best performance on four out of seven datasets, and on the remaining three datasets, the performance gap does not exceed 10%. Considering the substantial differences in parameter scale and inference speed between SVTime and these large pre-trained models, such comparable results demonstrate that directly training a lightweight model on the target dataset—without costly pre-training and fine-tuning—can still be highly effective and valuable.

Table 7: Full LTSF results on the benchmark datasets with Light-weight Models. Lower MSE and MAE indicate better forecasting accuracy. Red denote the best performance, blue indicate the second-best results.

	Pred_len	SVTIME-t MSE	SVTIME-t MAE	SVTIME MSE	SVTIME MAE	Dlinear MSE	Dlinear MAE	FITS MSE	FITS MAE	TexFilter MSE	TexFilter MAE	PaiFilter MSE	PaiFilter MAE	SparceTSF MSE	SparceTSF MAE	CycleNet MSE	CycleNet MAE	CMoS MSE	CMoS MAE
ETTh1	96	0.368	0.394	0.361	0.387	0.379	0.405	0.396	0.414	0.390	0.422	0.385	0.411	0.408	0.425	0.374	0.398	0.377	0.401
	192	0.406	0.418	0.409	0.408	0.448	0.461	0.430	0.435	0.418	0.437	0.421	0.433	0.438	0.447	0.404	0.417	0.410	0.422
	336	0.431	0.434	0.445	0.428	0.449	0.452	0.446	0.444	0.432	0.449	0.437	0.445	0.423	0.444	0.435	0.437	0.426	0.434
	720	0.464	0.472	0.457	0.461	0.512	0.519	0.448	0.462	0.457	0.477	0.474	0.481	0.433	0.461	0.464	0.472	0.451	0.466
ETTh2	96	0.283	0.344	0.283	0.329	0.280	0.347	0.286	0.349	0.303	0.364	0.303	0.361	0.311	0.362	0.308	0.339	0.306	0.363
	192	0.349	0.388	0.348	0.374	0.355	0.399	0.345	0.386	0.370	0.402	0.366	0.399	0.364	0.397	0.335	0.382	0.355	0.395
	336	0.379	0.416	0.364	0.395	0.435	0.453	0.364	0.403	0.391	0.424	0.389	0.426	0.373	0.408	0.369	0.413	0.367	0.407
	720	0.419	0.448	0.408	0.446	0.697	0.591	0.389	0.429	0.423	0.449	0.425	0.452	0.393	0.430	0.419	0.449	0.419	0.432
ETTm1	96	0.303	0.348	0.305	0.341	0.312	0.359	0.318	0.358	0.313	0.366	0.305	0.359	0.313	0.353	0.309	0.354	0.305	0.352
	192	0.337	0.368	0.320	0.351	0.350	0.383	0.348	0.375	0.352	0.389	0.340	0.380	0.342	0.370	0.350	0.377	0.341	0.371
	336	0.369	0.385	0.353	0.378	0.378	0.400	0.379	0.392	0.380	0.404	0.370	0.395	0.373	0.388	0.372	0.391	0.372	0.388
	720	0.422	0.414	0.407	0.405	0.439	0.441	0.431	0.420	0.439	0.438	0.439	0.427	0.423	0.415	0.426	0.421	0.433	0.423
ETTm2	96	0.165	0.254	0.182	0.253	0.168	0.261	0.169	0.259	0.188	0.276	0.179	0.265	0.166	0.256	0.166	0.254	0.177	0.264
	192	0.224	0.293	0.243	0.304	0.227	0.306	0.223	0.295	0.250	0.317	0.235	0.308	0.223	0.293	0.220	0.292	0.233	0.302
	336	0.280	0.331	0.294	0.341	0.281	0.342	0.275	0.329	0.314	0.358	0.281	0.338	0.273	0.326	0.281	0.334	0.286	0.336
	720	0.369	0.387	0.339	0.387	0.381	0.407	0.363	0.383	0.400	0.409	0.364	0.392	0.362	0.382	0.366	0.388	0.371	0.389
Weather	96	0.156	0.207	0.169	0.223	0.173	0.236	0.174	0.228	0.154	0.208	0.145	0.198	0.173	0.227	0.170	0.224	0.158	0.209
	192	0.198	0.245	0.214	0.262	0.220	0.282	0.216	0.263	0.199	0.248	0.190	0.240	0.215	0.261	0.212	0.260	0.199	0.247
	336	0.250	0.286	0.257	0.293	0.262	0.316	0.262	0.296	0.249	0.285	0.241	0.278	0.261	0.295	0.258	0.293	0.248	0.285
	720	0.321	0.336	0.321	0.340	0.325	0.367	0.327	0.343	0.328	0.339	0.316	0.332	0.327	0.343	0.323	0.339	0.319	0.334
Electricity	96	0.135	0.231	0.127	0.221	0.136	0.234	0.273	0.385	0.132	0.231	0.132	0.230	0.168	0.266	0.128	0.223	0.137	0.231
	192	0.151	0.245	0.143	0.234	0.151	0.248	0.284	0.392	0.155	0.251	0.150	0.246	0.188	0.285	0.144	0.237	0.153	0.246
	336	0.167	0.261	0.158	0.248	0.166	0.266	0.298	0.401	0.171	0.270	0.167	0.263	0.201	0.298	0.160	0.256	0.166	0.259
	720	0.207	0.295	0.198	0.283	0.200	0.298	0.331	0.420	0.209	0.304	0.209	0.299	0.229	0.319	0.198	0.288	0.206	0.293
Traffic	96	0.395	0.275	0.353	0.231	0.398	0.282	0.411	0.298	0.379	0.280	0.385	0.277	0.411	0.285	0.389	0.275	0.401	0.274
	192	0.408	0.281	0.364	0.243	0.410	0.287	0.420	0.298	0.407	0.296	0.406	0.288	0.425	0.295	0.403	0.281	0.414	0.279
	336	0.418	0.285	0.375	0.248	0.420	0.293	0.429	0.302	0.419	0.302	0.417	0.294	0.429	0.296	0.410	0.285	0.424	0.286
	720	0.454	0.305	0.421	0.271	0.457	0.314	0.466	0.322	0.453	0.318	0.457	0.317	0.465	0.315	0.446	0.305	0.460	0.305
Solar	96	0.206	0.257	0.183	0.238	0.206	0.279	0.778	0.706	0.189	0.253	0.184	0.240	0.213	0.257	0.204	0.273	0.204	0.252
	192	0.229	0.271	0.213	0.243	0.227	0.293	0.777	0.705	0.215	0.280	0.199	0.255	0.233	0.266	0.225	0.289	0.222	0.259
	336	0.242	0.281	0.228	0.243	0.242	0.304	0.775	0.705	0.210	0.271	0.204	0.260	0.248	0.275	0.239	0.293	0.232	0.269
	720	0.250	0.285	0.228	0.254	0.251	0.311	0.766	0.703	0.220	0.278	0.209	0.264	0.254	0.276	0.248	0.301	0.237	0.270
Wins		2 (17)		39 (2)		1 (0)		2 (7)		0 (5)		11 (5)		5 (5)		4 (16)		0 (9)	

Table 8: Full LTSF results on the benchmark datasets with Complex Models. Lower MSE and MAE indicate better forecasting accuracy. Red denote the best performance, blue indicate the second-best results.

	Pred_len	SVTIME-t MSE	SVTIME-t MAE	SVTIME MSE	SVTIME MAE	TimeMixer MSE	TimeMixer MAE	TiDE MSE	TiDE MAE	SCINet MSE	SCINet MAE	TimesNet MSE	TimesNet MAE	Autoformer MSE	Autoformer MAE	FEDFormer MSE	FEDFormer MAE	PatchTST MSE	PatchTST MAE	iTransformer MSE	iTransformer MAE
ETTh1	96	0.368	0.394	0.361	0.387	0.397	0.424	0.378	0.402	0.397	0.418	0.453	0.462	0.500	0.499	0.420	0.462	0.381	0.410	0.403	0.426
	192	0.406	0.418	0.409	0.408	0.448	0.458	0.413	0.424	0.529	0.488	0.500	0.494	0.516	0.519	0.474	0.498	0.420	0.437	0.435	0.448
	336	0.431	0.434	0.445	0.428	0.495	0.491	0.436	0.438	0.476	0.470	0.515	0.507	0.510	0.516	0.480	0.492	0.440	0.452	0.451	0.462
	720	0.464	0.472	0.457	0.461	0.538	0.528	0.456	0.467	0.529	0.513	0.686	0.593	0.650	0.607	0.547	0.540	0.496	0.508	0.570	0.546
ETTh2	96	0.283	0.344	0.283	0.329	0.292	0.357	0.277	0.339	0.345	0.386	0.353	0.405	0.412	0.462	0.392	0.450	0.278	0.340	0.307	0.363
	192	0.349	0.388	0.348	0.374	0.362	0.403	0.342	0.383	0.409	0.427	0.383	0.422	0.425	0.471	0.429	0.474	0.342	0.381	0.375	0.405
	336	0.379	0.416	0.364	0.395	0.378	0.420	0.363	0.404	0.407	0.439	0.399	0.438	0.415	0.465	0.430	0.481	0.371	0.408	0.441	0.446
	720	0.419	0.448	0.408	0.446	0.379	0.426	0.389	0.431	0.433	0.458	0.454	0.473	0.499	0.522	0.498	0.516	0.395	0.432	0.459	0.475
ETTm1	96	0.303	0.348	0.305	0.341	0.306	0.360	0.312	0.354	0.345	0.383	0.368	0.395	0.576	0.504	0.382	0.435	0.294	0.348	0.314	0.367
	192	0.337	0.368	0.320	0.351	0.388	0.405	0.345	0.373	0.403	0.418	0.417	0.417	0.571	0.509	0.398	0.440	0.334	0.372	0.350	0.388
	336	0.369	0.385	0.353	0.378	0.442	0.440	0.377	0.392	0.436	0.439	0.495	0.458	0.579	0.511	0.445	0.464	0.371	0.396	0.380	0.405
	720	0.422	0.414	0.407	0.405	0.578	0.506	0.431	0.421	0.523	0.486	0.505	0.479	0.551	0.497	0.503	0.492	0.414	0.426	0.444	0.444
ETTm2	96	0.165	0.254	0.182	0.253	0.182	0.271	0.167	0.255	0.182	0.275	0.210	0.290	0.281	0.354	0.278	0.354	0.169	0.259	0.181	0.274
	192	0.224	0.293	0.243	0.304	0.256	0.319	0.220	0.291	0.245	0.318	0.278	0.334	0.305	0.366	0.311	0.375	0.223	0.298	0.243	0.315
	336	0.280	0.331	0.294	0.341	0.347	0.380	0.276	0.329	0.316	0.365	0.354	0.375	0.343	0.386	0.349	0.393	0.282	0.339	0.289	0.344
	720	0.369	0.387	0.339	0.387	0.461	0.445	0.364	0.385	0.441	0.431	0.452	0.434	0.432	0.450	0.434	0.447	0.362	0.389	0.374	0.396
Weather	96	0.156	0.207	0.169	0.223	0.156	0.215	0.171	0.224	0.171	0.227	0.172	0.227	0.296	0.359	0.298	0.366	0.149	0.200	0.164	0.217
	192	0.198	0.245	0.214	0.262	0.202	0.253	0.214	0.260	0.223	0.271	0.233	0.279	0.335	0.386	0.341	0.399	0.193	0.241	0.205	0.253
	336	0.250	0.286	0.257	0.293	0.263	0.299	0.260	0.294	0.283	0.315	0.317	0.336	0.388	0.420	0.397	0.440	0.245	0.283	0.253	0.289
	720	0.321	0.336	0.321	0.340	0.354	0.354	0.327	0.342	0.359	0.362	0.377	0.376	0.401	0.425	0.427	0.447	0.316	0.334	0.318	0.336
Electricity	96	0.135	0.231	0.127	0.221	0.151	0.256	0.136	0.232	0.193	0.306	0.186	0.291	0.273	0.374	0.219	0.333	0.133	0.229	0.133	0.229
	192	0.151	0.245	0.143	0.234	0.167	0.272	0.150	0.244	0.198	0.307	0.194	0.294	0.256	0.358	0.228	0.341	0.150	0.246	0.152	0.250
	336	0.167	0.261	0.158	0.248	0.196	0.297	0.167	0.261	0.221	0.328	0.239	0.330	0.267	0.368	0.228	0.341	0.167	0.263	0.167	0.266
	720	0.207	0.295	0.198	0.283	0.225	0.320	0.206	0.294	0.257	0.353	0.236	0.328	0.307	0.399	0.248	0.357	0.204	0.295	0.192	0.288
Traffic	96	0.395	0.275	0.353	0.231	0.416	0.310	0.407	0.297	0.492	0.394	0.599	0.325	0.631	0.393	0.591	0.368	0.369	0.257	0.363	0.262
	192	0.408	0.281	0.364	0.243	0.435	0.318	0.423	0.305	0.496	0.393	0.618	0.333	0.684	0.424	0.600	0.366	0.384	0.264	0.390	0.281
	336	0.418	0.285	0.375	0.248	0.439	0.320	0.433	0.311	0.509	0.398	0.617	0.336	0.680	0.410	0.614	0.374	0.396	0.271	0.401	0.287
	720	0.454	0.305	0.421	0.271	0.463	0.330	0.480	0.341	0.545	0.413	0.656	0.347	0.671	0.403	0.633	0.385	0.436	0.294	0.440	0.307
Solar	96	0.206	0.257	0.183	0.238	0.215	0.268	0.210	0.258	0.210	0.295	0.191	0.270	0.679	0.604	0.351	0.446	0.168	0.237	0.189	0.247
	192	0.229	0.271	0.213	0.243	0.213	0.274	0.231	0.271	0.228	0.314	0.214	0.288	0.981	0.760	0.341	0.432	0.184	0.246	0.206	0.272
	336	0.242	0.281	0.228	0.243	0.240	0.287	0.247	0.277	0.238	0.318	0.216	0.285	0.842	0.690	0.300	0.388	0.192	0.253	0.220	0.285
	720	0.250	0.285	0.228	0.254	0.232	0.289	0.258	0.293	0.234	0.314	0.244	0.303	0.888	0.716	0.328	0.400	0.204	0.263	0.211	0.272
Wins		3 (23)		35 (6)		2 (0)		9 (11)		0 (0)		0 (1)		0 (0)		0 (0)		14 (20)		1 (8)	

Table 9: Full Ablation analysis results of SVTime (-t). Lower MSE and MAE are better.

Method	Length	ETTh1 MSE	ETTh1 MAE	ETTh2 MSE	ETTh2 MAE	ETTm1 MSE	ETTm1 MAE	ETTm2 MSE	ETTm2 MAE	Weather MSE	Weather MAE	Electricity MSE	Electricity MAE	Traffic MSE	Traffic MAE
SVTime	96	0.361	0.387	0.283	0.329	0.305	0.341	0.182	0.253	0.169	0.223	0.127	0.221	0.353	0.231
	192	0.409	0.408	0.348	0.374	0.320	0.351	0.243	0.304	0.214	0.262	0.143	0.234	0.364	0.243
	336	0.445	0.428	0.364	0.395	0.353	0.378	0.294	0.341	0.257	0.293	0.158	0.248	0.375	0.248
	720	0.457	0.461	0.408	0.446	0.407	0.405	0.339	0.387	0.321	0.340	0.198	0.283	0.421	0.271
	Avg.	0.418	0.421	0.351	0.386	0.346	0.369	0.265	0.321	0.240	0.280	0.157	0.247	0.378	0.248
(a) - IB2	96	0.371	0.394	0.274	0.338	0.301	0.346	0.164	0.254	0.17	0.221	0.136	0.231	0.414	0.301
	192	0.405	0.416	0.335	0.378	0.339	0.368	0.22	0.291	0.214	0.26	0.151	0.244	0.489	0.336
	336	0.437	0.436	0.367	0.406	0.372	0.387	0.273	0.33	0.26	0.294	0.166	0.26	0.513	0.368
	720	0.467	0.468	0.391	0.432	0.43	0.419	0.357	0.382	0.326	0.341	0.205	0.292	0.523	0.371
	Avg.	0.420	0.429	0.342	0.389	0.361	0.380	0.254	0.314	0.243	0.279	0.165	0.257	0.485	0.344
(b) - Backcast	96	0.367	0.389	0.309	0.352	0.298	0.344	0.172	0.262	0.198	0.259	0.186	0.270	0.454	0.254
	192	0.412	0.416	0.360	0.390	0.333	0.363	0.230	0.299	0.243	0.292	0.202	0.283	0.519	0.281
	336	0.427	0.431	0.376	0.409	0.506	0.494	0.331	0.375	0.289	0.322	0.217	0.298	0.534	0.289
	720	0.439	0.450	0.433	0.457	0.544	0.511	0.416	0.424	0.323	0.333	0.253	0.327	0.566	0.304
	Avg.	0.411	0.421	0.369	0.402	0.420	0.428	0.287	0.340	0.263	0.301	0.214	0.295	0.518	0.282
SVTime-t	96	0.368	0.394	0.283	0.344	0.303	0.348	0.165	0.254	0.156	0.207	0.135	0.231	0.395	0.275
	192	0.406	0.418	0.349	0.388	0.337	0.368	0.224	0.293	0.198	0.245	0.151	0.245	0.408	0.281
	336	0.431	0.434	0.379	0.416	0.369	0.385	0.280	0.331	0.250	0.286	0.167	0.261	0.418	0.285
	720	0.464	0.472	0.419	0.448	0.422	0.414	0.369	0.387	0.321	0.336	0.207	0.295	0.454	0.305
	Avg.	0.417	0.430	0.357	0.399	0.358	0.379	0.259	0.316	0.231	0.269	0.165	0.258	0.419	0.286
(c) - IB2	96	0.385	0.405	0.301	0.369	0.318	0.352	0.218	0.289	0.249	0.286	0.143	0.237	0.404	0.275
	192	0.405	0.410	0.352	0.395	0.340	0.364	0.258	0.316	0.270	0.300	0.155	0.247	0.414	0.278
	336	0.432	0.426	0.382	0.425	0.368	0.381	0.301	0.344	0.301	0.320	0.170	0.263	0.423	0.283
	720	0.468	0.472	0.431	0.456	0.420	0.412	0.380	0.394	0.350	0.354	0.210	0.296	0.459	0.302
	Avg.	0.422	0.428	0.366	0.411	0.361	0.377	0.289	0.336	0.293	0.315	0.170	0.261	0.425	0.284
(d) - IB3	96	0.385	0.415	0.303	0.367	0.318	0.353	0.218	0.290	0.170	0.222	0.143	0.237	0.404	0.274
	192	0.424	0.437	0.362	0.389	0.341	0.365	0.258	0.316	0.215	0.260	0.155	0.247	0.414	0.277
	336	0.449	0.441	0.380	0.408	0.368	0.382	0.301	0.344	0.260	0.294	0.170	0.263	0.422	0.282
	720	0.452	0.469	0.393	0.432	0.421	0.412	0.380	0.393	0.326	0.342	0.209	0.296	0.458	0.299
	Avg.	0.428	0.440	0.359	0.399	0.362	0.378	0.289	0.336	0.243	0.280	0.169	0.260	0.425	0.283
(e) - Backcast	96	0.435	0.463	0.335	0.377	0.545	0.434	0.248	0.314	0.25	0.29	0.18	0.265	0.564	0.371
	192	0.471	0.482	0.37	0.398	0.553	0.441	0.282	0.335	0.273	0.304	0.199	0.282	0.623	0.386
	336	0.513	0.543	0.375	0.407	0.568	0.451	0.321	0.359	0.303	0.324	0.215	0.296	0.634	0.389
	720	0.532	0.552	0.404	0.433	0.593	0.471	0.401	0.408	0.352	0.359	0.252	0.329	0.668	0.404
	Avg.	0.488	0.510	0.371	0.404	0.565	0.449	0.313	0.354	0.295	0.319	0.212	0.293	0.622	0.387

Table 10: The comparison with Pretrained time series models. Red denotes performance superior to SVTime.

Model	ETTh1 MSE	ETTh1 MAE	ETTh2 MSE	ETTh2 MAE	ETTm1 MSE	ETTm1 MAE	ETTm2 MSE	ETTm2 MAE	Weather MSE	Weather MAE	Electricity MSE	Electricity MAE	Traffic MSE	Traffic MAE
SVTime	0.418	0.421	0.351	0.386	**0.346**	**0.369**	**0.265**	**0.321**	0.240	0.280	**0.157**	**0.247**	**0.378**	**0.248**
CALF (2024)	0.432	0.431	0.333	**0.369**	0.368	0.385	0.355	0.365	0.280	0.304	0.176	0.266	0.421	0.274
Improvement	+3%	+2%	-5%	-4%	+6%	+4%	+34%	+14%	+17%	+9%	+12%	+8%	+11%	+10%
GPT4TS (2023)	0.418	0.421	0.336	0.373	0.350	0.381	0.323	0.350	0.251	0.288	0.170	0.263	0.421	0.274
Improvement	0%	+0%	-4%	-3%	+1%	+3%	+22%	+9%	+5%	+3%	+8%	+7%	+11%	+10%
TimeLLM (2024)	0.418	0.432	0.346	0.384	0.346	0.381	0.318	0.349	0.265	0.299	0.165	0.259	0.422	0.281
Improvement	0%	+3%	-1%	-1%	0%	+3%	+20%	+9%	+10%	+7%	+5%	+5%	+11%	+13%
VisionTS (2025)	0.409	**0.417**	0.359	0.391	0.345	0.373	0.269	0.328	**0.224**	**0.257**	0.161	0.253	0.387	0.255
Improvement	-2%	-1%	+2%	+1%	0%	+1%	+2%	+2%	-7%	-8%	+3%	+2%	+2%	+3%
TimeVLM (2025)	0.405	0.420	**0.317**	0.371	0.349	0.387	0.309	0.348	0.247	0.292	0.172	0.272	0.419	0.304
Improvement	-3%	0%	-10%	-4%	+1%	+5%	+17%	+8%	+3%	+5%	+10%	+10%	+11%	+22%
LightGTS (2025)	**0.394**	0.419	0.323	0.383	0.346	0.379	0.393	0.383	0.257	0.283	0.187	0.279	0.439	0.329
Improvement	-6%	-1%	-8%	-1%	0%	+3%	+49%	+19%	+7%	+1%	+19%	+13%	+16%	+32%