From Images to Signals: Are Large Vision Models Useful for Time Series Analysis?

Ziming Zhao[1], **ChengAo Shen**[1], **Hanghang Tong**[2], **Dongjin Song**[3], **Zhigang Deng**[1], **Qingsong Wen**[4], **Jingchao Ni**[1]
[1]University of Houston, [2]University of Illinois at Urbana-Champaign,
[3]University of Connecticut, [4]Squirrel Ai Learning
[1]{zzhao35,cshen9,zdeng4,jni7}@uh.edu, [2]htong@illinois.edu,
[3]dongjin.song@uconn.edu, [4]qingsongedu@gmail.com

Abstract

Transformer-based models have gained increasing attention in time series research, driving interest in Large Language Models (LLMs) and foundation models for time series analysis. As the field moves toward multi-modality, Large Vision Models (LVMs) are emerging as a promising direction. In the past, the effectiveness of Transformer and LLMs in time series has been debated. When it comes to LVMs, a similar question arises: are LVMs truely useful for time series analysis? To address it, we design and conduct the first principled study involving 4 LVMs, 8 imaging methods, 18 datasets and 26 baselines across both high-level (classification) and low-level (forecasting) tasks, with extensive ablation analysis. Our findings indicate LVMs are indeed useful for time series classification but face challenges in forecasting. Although effective, the contemporary best LVM forecasters are limited to specific types of LVMs and imaging methods, exhibit a bias toward forecasting periods, and have limited ability to utilize long look-back windows. We hope our findings could serve as a cornerstone for future research on LVM- and multimodal-based solutions to different time series tasks.

1 Introduction

Time series analysis is useful across various domains, including geoscience [1], neuroscience [4], energy [24], healthcare [38], and smart city [37]. With the significant advances of sequence modeling in the language domain, recent research attention on time series has been drawn to methods ranging from Transformer [48] to Large Language Models (LLMs) [18, 60, 20]. As Large Vision Models (LVMs), such as ViT [9], BEiT [3] and MAE [15], became successful, some emergent efforts have been invested to explore the potential of LVMs in time series modeling [5]. In these works, time series are *imaged*, *i.e.*, transformed to certain image representations [40], as illustrated by Fig. 1(a), then fed to an LVM to learn embeddings that can be probed for downstream tasks. These works posit that LVMs, being pre-trained on vast images, are useful in time series analysis from two perspectives: (1) for *high-level* (*i.e.*, semantic level) tasks such as classification, imaged time series can encode distinguishable temporal patterns as semantic cues that LVMs can recognize; (2) for *low-level* (*i.e.*, numerical level) tasks such as forecasting, the intrinsic relationship between images and time series – each row/column in an image (per channel) is a sequence of *continuous* pixel values that resembles a univariate time series (UTS) – makes LVMs better suited to time series tasks than LLMs since LLMs consume *discrete* tokens. However, in-depth connections between LVMs and time series analysis remain largely underexplored.

In the past several years, the effectiveness of Transformer and LLMs for time series analysis was critically questioned by [55] and [44] in tandem. When it comes to LVMs, a similar question arises – *are LVMs useful for time series analysis?* To underlie future research upon LVMs, including

Preprint. Under review.

multi-modal models that integrate imaged time series [61], a thorough feasibility study is needed to understand LVMs' role in time series tasks. This is precisely our goal. We comprehensively study LVMs on two representative tasks, time series classification (TSC) and time series forecasting (TSF). In a nutshell, our conclusion is cautiously positive: **pre-trained LVMs are useful in TSC but pose challenges when used for TSF.** The current best LVM-based forecasters, although effective, are limited to specific types of LVMs and imaging methods, exhibit bias towards forecasting periods, and have limited ability to utilize long look-back window.

In this work, we choose two LVMs that are supervisedly pre-trained, *i.e.*, ViT [9] and Swin [36], and two LVMs that are self-supervisedly pre-trained, *i.e.*, MAE [15] and SimMIM [53], along with 8 widely used methods for imaging time series as suggested in [40]. Our analysis involves 10 datasets for TSC and 8 datasets for TSF, all are widely used benchmarks [2, 51, 64, 41, 55, 44]. In §3, we introduce methods for adapting LVMs to time series tasks, including *input alignment* and *task-specific designs.* Our analysis (§4) starts with thorough comparisons between LVMs and the state-of-the-art (SOTA) baselines, including 18 classification baselines and 8 forecasting baselines. The results provide an overview on the effectiveness of LVMs, shedding light on what type of LVMs (*supervised vs. self-supervised*), which imaging method (*among 8 methods*), and what output design (*linear probing vs. pre-trained decoder*) fit which task (*classification vs. forecasting*).

To figure out the source of effectiveness, we compare LVMs' zero-shot and (fully/partially) fine-tuned performance with that of the same architecture trained from scratch, which shows the pre-trained Transformer components indeed transfer useful knowledge. By testing LVMs under different shuffling of time steps, we also find that LVMs grasp sequence modeling capability. As we observe TSF is more challenging than TSC to LVMs, further TSF-specific study is conducted. From it, we reveal the best LVM forecaster is a combination of self-supervised LVMs and a specific imaging method (*i.e.*, UVH in Fig. 1(a)). Moreover, the pre-trained decoders in self-supervised LVMs play a more critical role than their encoders in forecasting. However, the current best LVM forecasters have an inductive bias that renders them basically "combine past periods" as forecasts, thus they are prone to datasets with strong periodicity. To sum up, our contributions are as follows:

- To the best of our knowledge, this is the first work to comprehensively study the feasibility of LVMs in time series analysis for both high-level and low-level predictive tasks.
- We compare representative LVMs using different imaging methods on datasets of various domains, summarize the current best ways to tweak LVMs for TSC and TSF tasks, assess various aspects of the adapted LVMs, including their effectiveness in terms of pre-training, imaging, decoding, fine-tuning, architecture, temporal order of data, and computational costs, for the two tasks.
- We further investigate the challenge of using LVMs for forecasting by studying individual model components, potential inductive bias, and the impact of look-back windows.

We hope our findings could provide an in-depth insight of LVMs' role in time series analysis, so as to benefit future development in this emergent area and multi-modal time series research [17, 31].

2 Related Work

Our work share similar merits as [55, 44, 65], each of which sheds important lights on a single time series task, *i.e.*, Transformers for TSF [55], LLMs for TSF [44], and LLMs for time series anomaly detection (TSAD) [65]. In contrast, our work is LVM-specific, covering more tasks with in-depth analysis. This work could be considered as a substantial complement to the prior works by adding a new lens to our understanding of large models' roles in the contemporary time series domain.

Vision models have been used for a variety of time series tasks, including classification [29, 51], forecasting [56, 54], anomaly detection [58, 51], and generation [28, 21]. Our work focuses on the recent development of using **pre-trained LVMs**, particularly Transformer-based models, for time series analysis. Image-pretrained CNNs have also been investigated in the past, such as pre-trained ResNet for TSAD [39] and Inception-v1 for TSF [27], but are out of our scope due to their relatively smaller sizes. To apply LVMs to time series, the existing works typically employ one of the 8 imaging methods as summarized by [40], which we will introduce in §3 (Fig. 1(a)). For example, AST [11] applies ImageNet-pretrained DeiT [45] on filterbank spectrograms of audio signals, which are basically UTS, for TSC. ViTST [29] uses pre-trained Swin [36] for classifying lineplots of time series. These works have inspired a series of efforts in pre-training ViT architectures with imaged time series data, such as SSAST on AudioSet-2M [12], ViTime on synthetic data [54], and Brain-JEPA on brain

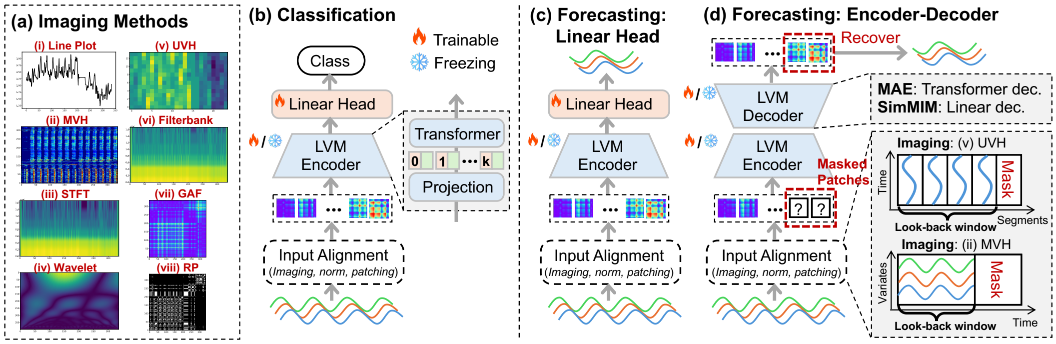

Figure 1: An overview of (a) different imaging methods, (b) LVM-based time series classification, (c) LVM with linear head for forecasting, (d) LVM encoder-decoder for forecasting. In (a), MVH encodes MTS, others encode UTS. (b)(c) apply to all LVMs (ViT, Swin, MAE, SimMIM) in this study. (d) applies to MAE and SimMIM with UVH/MVH images. Table 1 summarizes their applicability.

time series [8]. In contrast to TSC, TSF task has less efforts in using LVMs, possibly because LVMs are less adept at low-level tasks than high-level tasks. The most salient method is VisionTS [5], which adapts a self-supervisedly pre-trained LVM *i.e.*, MAE [15], to zero-shot and few-shot TSF. In our work, in addition to MAE, we include another self-supervised LVM – SimMIM [53].

More recently, large vision-language models (VLMs), such as LLaVA [32], CLIP [43], ViLT [22], *etc.*, which involve pre-trained large vision encoders, have been explored for TSC [49, 42], TSAD [66], and TSF [61]. However, the effectiveness of sole LVMs in time series analysis has not yet been well understood. As such, we focus on LVMs in this work, and leave VLMs for future work. We refer readers to [40] for a detailed discussion about the existing literature on LVMs for time series.

3 Methods for Using LVMs in Time Series Analysis

Following existing LVM-based solutions [5], we assess LVMs' innate ability in time series analysis by keeping the main architecture intact but making a few necessary tweaks, including (**i**) input alignment; and (**ii**) task-specific augmentation. Additionally, we introduce two ablations that will be used in §4 to evaluate whether LVMs' architecture is over-complex.

Input Alignment. The input to a pre-trained LVM should be a normalized 3-channel image of a predefined size. Fitting time series to LVMs' input requires (1) imaging time series; (2) resizing the imaged time series to fit the channel/size requirement; and (3) normalizing the image.

For (1), we employ 8 imaging methods suggested by [40]. As illustrated in Fig. 1(a), They include Line Plot, multivariate heatmap (MVH), univariate heatmap (UVH), Short-Time Fourier Transform (STFT), Wavelet Transform, Filterbank, Gramian Angular Field (GAF), and Recurrence Plot (RP). **Line Plot** is a straightforward method that draws a 2D image with x-axis representing time steps and y-axis representing time-wise values. **MVH** visualizes the matrix of a multivariate time series (MTS), $\mathbf{X} \in \mathbb{R}^{d \times T}$, with x-axis representing T time steps and y-axis representing d variates. **UVH** is a method proposed by TimesNet [51] and used by other methods [5, 30]. It divides a UTS, $\mathbf{x} \in \mathbb{R}^{T}$, into $\lfloor T/L \rfloor$ segments of length L, where L is a period obtained using Fast Fourier Transform (FFT) on $\mathbf{x}$. The segments are then stacked to a 2D image of size $L \times \lfloor T/L \rfloor$. **STFT**, **Wavelet** and **Filterbank** are three methods for transforming $\mathbf{x}$ to a spectrogram with x-axis representing time and y-axis representing frequency. **GAF** and **RP** produce square matrices with both x- and y-axis representing time, but they encode different temporal patterns. Among the 8 methods, MVH encodes MTS, while others encode UTS, leading to different ways to model multiple variates as stated in "Task-Specific Augmentation". We refer readers to [40] for more details about the 8 imaging methods.

For (2), *i.e.*, image resizing, following [11, 5], we first resize an imaged time series to fit the size defined by LVMs' pre-training data using bilinear interpolation. Then, we align the resized images to meet the 3-channel requirement by duplicating each resized image (per variate) three times to form a gray image. For (3), *i.e.*, image normalization, since the adopted LVMs, *i.e.*, ViT, Swin, MAE, SimMIM, standardize each pre-training image, we normalize each imaged time series in the same manner for consistency: $\mathbf{I}_{\text{norm}} = [\mathbf{I} - \text{mean}(\mathbf{I})]/\text{standard-deviation}(\mathbf{I})$, where $\mathbf{I}$ is the input image and $\mathbf{I}_{\text{norm}}$ is the normalized one. As shown in Fig. 1(b)-(d), the normalized image is then divided into a number of patches as specified by each LVM before feeding to the LVM.

Task-Specific Augmentation. For **TSC task**, as shown in Fig. 1(b), we linearly probe each LVM's encoder. For ViT and Swin, this implies replacing their classification layers by a new linear layer tailored to a specific downstream TSC task. For MAE and SimMIM, this means their reconstruction decoders are replaced by a linear classification layer. As most imaging methods encode UTS (except for MVH), the image of each variate is fed to the LVM individually. The output patch embeddings of all variates are concatenated before delivering to the last linear layer. For MVH, there is a single image of all variates, thus it does not need variate-concatenation.

For **TSF task**, we employ two frameworks from the literature. Fig. 1(c) trains a linear forecaster [56, 54], Fig. 1(d) uses LVMs' reconstruction decoders for forecasting [5]. Because only MAE and SimMIM in our study have such decoders, Fig. 1(d) is applied to them. Fig. 1(c) applies to ViT and Swin. For both frameworks, we adopt the "variate-independence" assumption that is widely used in TSF [41], *i.e.*, each variate is forecasted independently. This applies to all imaging methods except for MVH, by which all variates appear in the same image thus are forecasted once. Additionally, the framework in Fig. 1(d) adds a mask subsequent to the look-back window part in the image, then it reconstructs the masked patches and recovers forecasts. This requires input images to preserve raw time series values in pixels. Among the 8 imaging methods, only MVH and UVH preserve time series values. Thus, this framework is only applied to MVH and UVH. The framework in Fig. 1(c) can be applied to all imaging types. Table 1 summarizes how frameworks (b)(c)(d) in Fig. 1 apply to different LVMs.

Task	Imaging	ViT	Swin	MAE	SimMIM
Classification	All	(b)	(b)	(b)	(b)
Forecasting	UVH,MVH	(c)	(c)	(d)	(d)
Forecasting	Other	(c)	(c)	(c)	(c)

Table 1: LVM framework summary. (b)(c)(d) indicates the frameworks in Fig. 1.

Ablations. To assess whether LVM architecture is over-complex, we add two ablation models. Both models keep the projection layer in LVM encoder, but replace the Transformer by a simpler layer. The first ablation uses a linear layer, named as W/O-LVM. The second ablation uses a single randomly initialized multi-head attention layer, named as LVM2ATTN. Both ablations use a linear head to avoid complex decoders. They are applicable to all 8 imaging types and both of the two tasks. An illustration of the ablation models can be found in Appendix B.6.

4 Experiments

4.1 Experimental Setup

Datasets. Our experiments are conducted on widely used benchmarks. For TSC, following [51, 64], we use 10 datasets from UEA Archive [2], covering gesture/action/audio recognition, heartbeat-based diagnosis, and other real-world tasks. The datasets are preprocessed following [57]. For TSF, we use 8 datasets including ETT (Electricity Transformer Temperature) [62], encompassing ETTh1, ETTh2, ETTm1, ETTm2, Weather [52], Illiness [52], Traffic [52], and Electricity [46]. For both tasks, all of the time series are MTS. We defer detailed data descriptions to Appendix A.1.

Evaluation Metrics. For TSC, following [51, 64], we report classification accuracy of the compared methods. For TSF, following [41, 55, 44], mean squared error (MSE) and mean absolute error (MAE) are used to evaluate performance. Definitions of the metrics are deferred to Appendix A.3.

Models. We include two supervised LVMs: (1) ViT [9], (2) Swin [36], and two self-supervised LVMs: (3) MAE [15], (4) SimMIM [53]. They are implemented as per Table 1 for different tasks. Following [51, 64], we include 18 classification baselines ranging from XGBoost to LLMs. Following [44, 5], 8 SOTA forecasting baselines are compared. The baseline methods are presented in Fig. 2 and Table 2, and described in Appendix A.2. The implementation details of the LVMs, including checkpoint selection, hyperparameters, and running environments are in Appendix A.4.

4.2 Results of Comparing LVMs with Non-LVM Methods

Fig. 2 and Table 2 present the overall performance of the compared methods. In the comparisons, ViT and MAE are selected to represent LVMs for their best performance in their respective group: supervised LVM group and self-supervised LVM group. In §4.3, we will compare ViT, Swin, MAE

Method	MAE		ViT		Time-LLM		GPT4TS		CALF		Dlinear		PatchTST		TimesNet		FEDformer		Autoformer	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.409	0.419	0.445	0.449	0.418	0.432	0.418	0.421	0.432	0.431	0.423	0.437	0.413	0.431	0.458	0.450	0.440	0.460	0.496	0.487
ETTh2	0.357	0.390	0.389	0.411	0.361	0.396	0.354	0.389	0.351	0.384	0.431	0.447	0.330	0.379	0.414	0.427	0.437	0.449	0.450	0.459
ETTm1	0.345	0.374	0.409	0.422	0.356	0.377	0.363	0.378	0.396	0.391	0.357	0.379	0.351	0.381	0.400	0.406	0.448	0.452	0.588	0.517
ETTm2	0.268	0.327	0.300	0.337	0.261	0.316	0.254	0.311	0.283	0.323	0.267	0.334	0.255	0.315	0.291	0.333	0.305	0.349	0.327	0.371
Weather	0.225	0.258	0.234	0.273	0.244	0.270	0.227	0.255	0.251	0.274	0.249	0.300	0.226	0.264	0.259	0.287	0.309	0.360	0.338	0.382
Illness	1.837	0.883	2.179	1.016	2.018	0.894	1.871	0.852	1.700	0.869	2.169	1.041	1.443	0.798	2.139	0.931	2.847	1.144	3.006	1.161
Traffic	0.386	0.256	0.430	0.343	0.422	0.281	0.421	0.274	0.444	0.284	0.434	0.295	0.391	0.264	0.620	0.336	0.610	0.376	0.628	0.379
Electricity	0.159	0.250	0.173	0.266	0.165	0.259	0.170	0.263	0.176	0.266	0.166	0.264	0.162	0.253	0.193	0.295	0.214	0.327	0.227	0.338
# Wins	**9**		0		0		3		0		0		**4**		0		0		0	

Table 2: Model comparison in TSF. The results are averaged over different prediction lengths. See Table 11 in Appendix B.2 for full results. Red and Blue numbers are the the best and second best results. # Wins is the number of times the method performed best.

and SimMIM. Here, ViT and MAE are set up with their best imaging methods — GAF for TSC and UVH for TSF. Comparisons of different imaging methods are also discussed in §4.3. On average, LVMs were fine-tuned on each dataset with 20 epochs for TSC and 8 epochs for TSF upon early stopping. Our experiments follow standard protocols of TSC [64] and TSF [44]. In Fig. 2, we collected the results of the 18 baselines reproduced by [64]. In Table 2, the results of LLM based methods (*i.e.*, Time-LLM, GPT4TS, CALF) are reproduced by [44], the rest baseline results are reproduced by [5]. The full results can be found in Appendices B.1 and B.2.

From Fig. 2, both ViT and MAE outperform the baselines, which provides an overview of both supervised and self-supervised LVMs' potential in high-level (*i.e.*, semantic level) TSC task. This is consistent with their ability in classifying regular images [15]. From Table 2, across 8 datasets and 2 metrics, MAE outperforms non-LVM baselines in 9/16 cases, while ViT doesn't show evident superiority over non-LVM baselines, which may be caused by its classification-based pre-training. The results suggest LVMs' distinct abilities in TSF, conveying that more challenges may appear in low-level (*i.e.*, numerical level) tasks. Taking a closer look at Table 2, despite ViT's inferior performance, it is comparable to DLinear in many cases. It implies that although ViT is pre-trained for image classification, linearly probing it is adequate to produce reasonable forecasting results, showing a potential in cross-task/modality knowledge transfer.

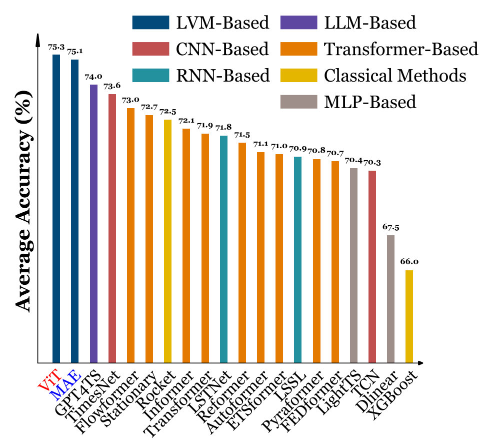

Figure 2: Model comparison in TSC. The results are averaged over 10 UEA datasets. See Table 9 in Appendix B.1 for full results.

4.3 In-Depth Analysis of LVMs' Suitability in Time Series Tasks

Next, we dissect LVMs' performance by answering a series of research questions. The following analyses use 4 UEA classification datasets (FaceDetection, Handwriting, SpokenArabicDigits, and UWaveGestureLibrary) and 4 forecasting datasets (ETTh1, ETTm1, Weather, and Illiness) for conciseness. Unless otherwise noted, the best-performing LVM is used for TSC, *i.e.*, ViT with GAF imaging (*ref.* Fig. 2), and TSF, *i.e.*, MAE with UVH imaging (*ref.* Table 2), respectively.

[RQ1] What type of LVM best fits TSC (TSF) task? Fig. 3 compares the 4 LVMs in TSC and TSF tasks. From Fig. 3, we observe (1) supervised LVMs and self-supervised LVMs show comparable accuracies in classification, while (2) self-supervised LVMs are remarkably better at forecasting than supervised LVMs. (1) is consistent with the comparable performance of the two kinds of LVMs in classifying images [15]. (2) attributes to the continuous nature of pixels and time series, which enables self-supervised LVMs to transfer their ability in reconstructing masked pixels to predict (masked) time series, as proposed by [5]. Moreover, in Fig. 3(a), we observe SimMIM and Swin underperform (SimMIM uses Swin backbone). This is because they use *window-based local attention* mechanism. Compared to the *global attention* used by MAE and ViT, local attention implicitly

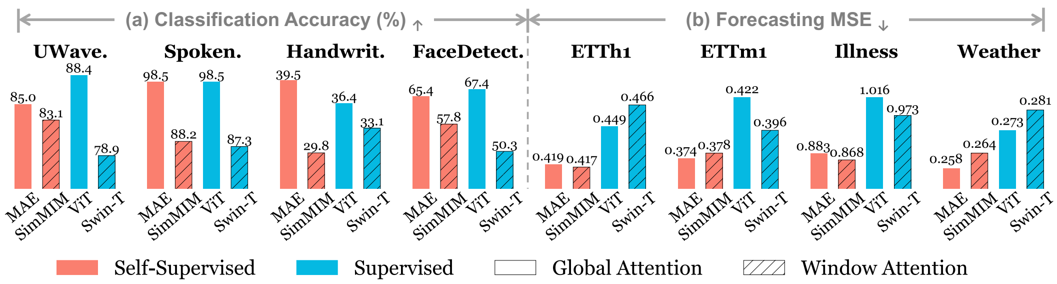

Figure 3: Comparison of 4 LVMs on TSC (accuracy) and TSF (MSE). ↑ (↓) indicates a higher (lower) value is better. Two taxonomies of the LVMs: (1) supervised (ViT, Swin) *vs.* self-supervised (MAE, SimMIM), (2) using global attention (ViT, MAE) *vs.* window-based attention (Swin, SimMIM).

Task		TSC Task (accuracy (%)$_{\uparrow}$)				TSF Task (MSE$_{\downarrow}$)			
Dataset		UWave.	Spoken.	Handwrit.	FaceDetect.	ETTh1	ETTm1	Illiness	Weather
RQ3	(a) All parameters	88.4	**98.5**	**36.4**	**67.4**	0.558	0.399	1.781	0.273
	(b) All but `CLS` & `Mask`	87.5	98.2	35.2	66.3	0.530	0.408	1.783	0.275
	(c) MLP & norm	**88.7**	98.4	35.5	67.1	0.532	0.396	1.737	0.264
	(d) Norm	81.6	98.0	28.5	65.2	**0.409**	**0.345**	1.837	**0.225**
	(e) Zero-shot	84.0	**98.5**	27.8	66.7	0.452	0.420	2.037	0.308
	(f) Train from scratch	73.4	97.0	24.3	65.0	0.475	0.372	**1.723**	0.241
RQ4	w/o-LVM	78.6	96.4	22.4	64.1	0.423	0.376	2.291	0.255
	LVM2Attn	80.1	96.5	20.7	66.2	0.428	0.357	2.108	0.254

Table 3: Ablation analysis of LVMs. For classification, higher accuracy indicates better performance. For forecasting, lower MSE is preferred. Full results are in Appendices B.5 and B.6.

assumes *translation invariance* – a model's ability to recognize an object in an image regardless of where the object appears [26]. This assumption, however, does not hold in imaged time series since different locations in an imaged time series correspond to different time-steps/frequencies, which are ordered. A pattern that appears at different time steps may lead to different classes. By overlooking spatial differences, SimMIM and Swin fail to identify some time/frequency-sensitive patterns.

[RQ2] Which imaging method best fits TSC (TSF) task?

Fig. 4 presents the critical difference (CD) diagrams [14] on the average rank of the 8 imaging methods on TSC and TSF tasks (lower rank is better). The detailed results are in Appendix B.4. From Fig. 4(a), GAF fits the classification best, with close performance to MVH and RP, indicating their abilities in encoding distinguishable semantic patterns. Line Plot remarkably underperforms, thus may not fit this task. For forecasting, UVH and MVH are used in conjunction with the reconstruction framework in Fig. 1(d) because they preserve raw time series values in pixels. Other imaging methods produce pixels with different meanings, rendering reconstruction inappropriate, thus they use the framework in Fig. 1(c). From Fig. 4(b), the best performance of UVH and MVH suggests their suitability in numerical level tasks by leveraging LVMs' knowledge acquired from reconstructing masked pixels during pre-training.

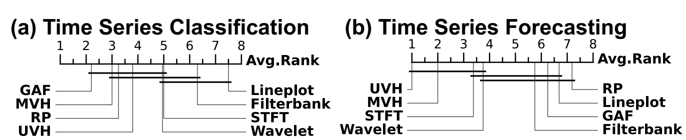

Figure 4: Average rank of different imaging methods in (a) TSC task, and (b) TSF task. Lower rank is better.

[RQ3] Are the pre-trained parameters in LVMs useful in time series tasks? We test whether the knowledge learned during pre-training is useful in time series tasks by comparing three kinds of ablations: (1) training LVMs from scratch, (2) freezing LVM's parameters (*i.e.*, zero-shot performance), and (3) fine-tuning LVMs with a few epochs. Since different tasks may need different fine-tuning strategies, we include a series of fine-tuning ablations that progressively freeze the key components in the Transformer block of LVMs. Fig. 5 shows the key components. To sum up, our ablations in this study include (a) Fine-tune all parameters; (b) Fine-tune all parameters but freeze `CLS` token and

Task		Classification				Forecasting			
Dataset		UWave.	Spoken.	Handwrit.	FaceDetect.	ETTh1	ETTm1	Illiness	Weather
Sf-All	w/o-LVM	78.2%	49.7%	81.7%	19.3%	76.2%	98.4%	116.4%	24.1%
	LVM2Attn	86.4%	50.6%	89.9%	22.4%	79.7%	117.1%	109.1%	24.4%
	LVM	80.7%	84.7%	91.5%	29.2%	83.8%	118.4%	162.8%	44.5%
Sf-Half	w/o-LVM	6.6%	12.4%	74.6%	10.8%	14.4%	28.3%	41.6%	2.4%
	LVM2Attn	8.7%	11.6%	83.6%	11.3%	19.5%	44.8%	69.3%	2.4%
	LVM	36.4%	30.2%	86.5%	9.3%	14.5%	48.2%	21.3%	9.6%
Ex-Half	w/o-LVM	98.8%	82.2%	83.5%	22.8%	13.0%	145.3%	11.0%	34.0%
	LVM2Attn	98.9%	82.3%	87.0%	24.6%	9.1%	158.3%	27.9%	35.5%
	LVM	59.4%	89.9%	97.0%	9.2%	14.2%	242.3%	23.0%	67.2%
Masking	w/o-LVM	-1.0%	3.1%	22.3%	-1.2%	47.3%	58.5%	94.1%	33.4%
	LVM2Attn	1.0%	3.6%	20.3%	2.7%	46.0%	70.3%	127.8%	33.6%
	LVM	29.0%	41.8%	56.0%	7.4%	47.5%	58.4%	128.9%	49.6%

Table 4: Performance drop of the compared models under different temporal perturbations. Red color marks the largest drop for each perturbation strategy. Full results are in Appendix B.7.

Mask token; (c) Fine-tune MLP and norm layers only; (d) Fine-tune norm layer only; (e) freeze all parameters (*i.e.*, zero-shot); and (f) randomly initialize an LVM and train it from scratch.

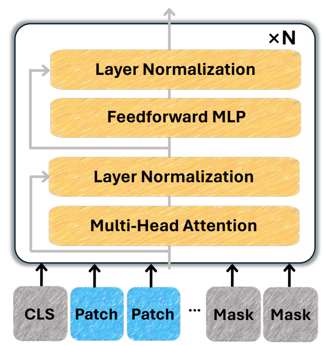

Figure 5: Key components in LVMs' Transformer block.

Table 3 (upper panel) summarizes the results. For TSC, we observe that zero-shot performance is better than training from scratch in all cases, suggesting LVMs indeed transfer useful knowledge. Fine-tuning all parameters with a few epochs always improves over zero-shot cases, further validating effective knowledge transfer. Moreover, fine-tuning MLP & norm layers is comparable to full fine-tuning, suggesting a minimal fine-tuning effort in this task. For TSF, surprisingly, neither of zero-shot case nor fine-tuning all parameters consistently outperforms training from scratch. However, only fine-tuning the norm layer significantly boosts the performance. This may be caused by the low-level nature of the forecasting task. The model needs to predict numerical values, which is largely influenced by normalization, while fine-tuning more than necessary may lead to overfitting. This is in contrast to classification, where the learning of high-level semantic patterns is influenced by more layers than normalization, thus fine-tuning more parameters is beneficial.

[RQ4] How useful are LVMs' architectures? In [RQ3], training LVMs from scratch is prone to overfitting due to LVMs' complex architectures. To examine whether LVMs' architecture is over-complex for time series analysis, we run the two simpler models introduced in §3, *i.e.*, w/o-LVM and LVM2Attn, which are less likely to overfit the training data. Table 3 (bottom panel) summarizes their results. We observe that training from scratch does not consistently outperform simple models. This implies that the LVM's architecture itself is over-complex. However, since training from scratch is no worse than the simpler models, the overfitting issue is not serious. Moreover, the zero-shot and all fine-tuning cases outperform w/o-LVM and LVM2Attn in TSC. Fine-tuning case (d) consistently outperforms w/o-LVM and LVM2Attn in TSF. These results indicate LVMs' architectures are not over-complex as a container of transferrable knowledge learned during pre-training.

[RQ5] Do LVMs capture temporal order of time series? Temporal order plays a critical role in time series analysis. Like [55] and [44], it is of significant interests to understand whether LVMs can capture the temporal information. To this end, following [44], we perturb the temporal order by four methods (1) **Sf-All**: randomly shuffle all of the time points; (2) **Sf-Half**: randomly shuffle the first half of the time points; (3) **Ex-Half**: swap the first and second halves of the time points; and (4) **Masking**: randomly mask 50% time points. Table 4 summarizes the relative performance drop. Following [55, 44], simple models are compared for their effectiveness in capturing temporal order. From Table 4, we can see that LVMs always have a performance drop under temporal perturbations. Moreover, they are more vulnerable to temporal perturbations than the ablations. This implies LVMs are very likely making effective use of temporal patterns in time series during their inferences.

[RQ6] What are the computational costs of LVMs? We evaluate the training and inference time of LVMs. Training time is measured when a model converges with early stopping. Inference time is

Method		LVM			1st Baseline (task specific)			2nd Baseline (task specific)		
Task	Dataset	# Param (M)	Train (min)	Inference(ms)	# Param (M)	Train (min)	Inference(ms)	# Param (M)	Time (min)	Inference(ms)
TSC	Handwrit.	97.59	5.18	23.72	83.62	1.33	50.51	2.47	0.51	0.78
	Spoken.	105.57	58.79	25.62	82.42	7.26	44.63	1.20	3.28	0.49
TSF	ETTh1	111.91	9.99	4.32	3.75	0.52	0.18	85.02	10.46	0.50
	Weather	111.91	207.83	1.50	6.90	16.97	0.10	86.64	94.10	0.35

Table 5: Computational costs of LVMs and two best baselines in TSC (GPT4TS, TimesNet) and TSF (PatchTST, GPT4TS). The forecasting costs are measured with prediction length 96.

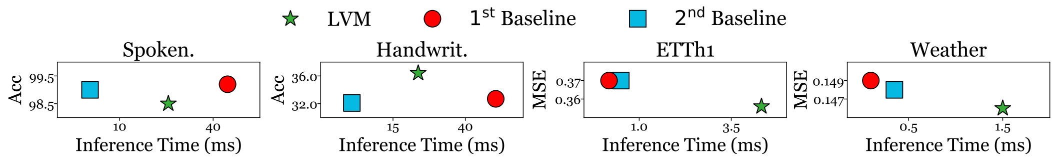

Figure 6: Inference time *vs.* performance of compared methods on TSC (accuracy) using Handwriting, SpokenArabicDigits, and TSF (MSE) using ETTh1, Weather. Full results are in Appendix B.10.

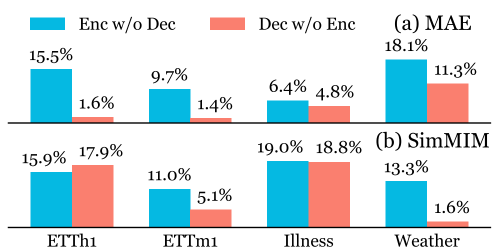

Figure 7: Forecasting performance drop (%) of (a) MAE and (b) SimMIM when only using encoder (blue) and decoder (red).

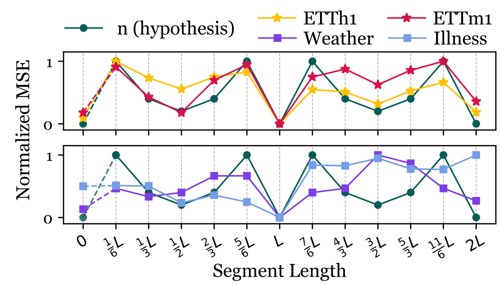

Figure 8: Forecasting performance of MAE *w.r.t.* varying segment length used in UVH imaging. n (green) estimates the difficulty of forecasting.

estimated by the average runtime per test sample. Table 5 compares LVMs with the best two baselines in TSC (Fig. 2) – GPT4TS, TimesNet, and TSF (Table 2) – PatchTST, GPT4TS. From Table 5, LVMs have more parameters than the baselines. On average, LVMs take 6x (15x) training time than the best TSC (TSF) baseline, primarily due to their larger sizes of trainable parameters. For inference, LVMs are 0.5x faster than the best TSC baseline, but are 24x slower than the best TSF baseline. This is incurred by both the parameter size and the extra costs to imaging time series. Fig. 6 shows inference time *vs.* performance. Compared to the best baselines, LVMs trade the computational overhead for better performance. This is also evident in Fig. 2 and Table 2. Considering the fast developing hardware, the results suggest a big potential of LVMs in future time series research.

4.4 More Analysis of LVMs' Suitability in Time Series Forecasting

As the forecasting task shows more challenges than the classification task, we conduct more in-depth analysis to dissect LVMs' potential in TSF as follows.

[RQ7] Which component of LVMs contributes more to forecasting? Usually, pre-trained encoders are considered as general feature extractors and widely used in knowledge transfer. In contrast, pre-trained decoders are task-specific thus are often abandoned in a downstream task. However, the conclusion looks counterintuitive when adapting LVMs to TSF. Fig. 7 shows the performance drop of two ablations relative to MAE and SimMIM: (1) **Enc w/o Dec** preserves the pre-trained encoder but randomly initializes the decoder; (2) **Dec w/o Enc** preserves the pre-trained decoder but randomly initializes the encoder. Both ablations are fine-tuned until convergence. From Fig. 7, for LVMs, **Enc w/o Dec** drops more than **Dec w/o Enc**, implying the **pre-trained decoders play more important roles than the encoders in TSF**. This is because LVMs' decoders aim to reconstruct pixel values, thus fitting the low-level TSF task. Surprisingly, SimMIM's decoder is merely a linear layer that only occupies 3.8% of all parameters, which however overwhelms its much larger encoder, further underscoring the essential role of LVMs' pre-trained decoders in forecasting.

[RQ8] Will period-based imaging method induce any bias? In Table 2, the best LVM forecaster is MAE with UVH imaging. As shown in Fig. 1(a)(d), UVH is a period-based imaging method –

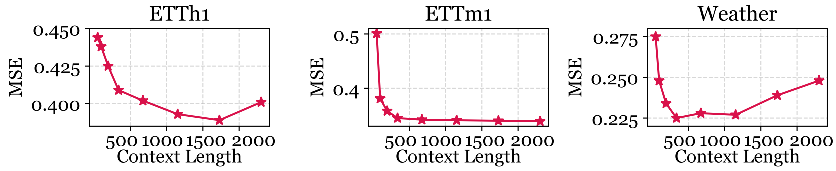

Figure 10: TSF performance (MSE) of MAE with varying look-back window (or context) lengths.

it stacks length-L segments of a UTS $\mathbf{x}$ into a 2D image of size $L \times \lfloor T/L \rfloor$, where L is a period. **We find this method leads to an inductive bias towards "forecasting periods".** In Fig. 8, we evaluate MAE's forecasting performance by changing the segment length from $\frac{1}{6}L$ to $\frac{12}{6}L$, where the MSE values are min-max normalized to range $[0, 1]$. In Fig. 8, an estimated MSE is added at 0 by averaging the MSEs at L and $2L$ since length-0 is not computable. This (and the green lines) will be used later. From Fig. 8, MAE's best performance occurs at L and $2L$, implying (1) the datasets show strong periodicity; and (2) MAE tends to infer future by "combining" past segments. When past segments do not coincide with periods, *i.e.*, $\neq L$ or $2L$, MAE fails to forecast accurately.

Interestingly, following the UVH imaging method, we can estimate the difficulty of TSF for MAE by using the segment length. Basically, the difficulty highly correlates with *how long a segment can reoccur*, measured by the number of segments between the two occurrences (Fig. 9). If the two occurrences are far apart, it is more difficult for MAE to capture periodic patterns. More formally, if we divide the UTS into length-$\frac{i}{k}L$ segments, *e.g.*, in Fig. 8, $k = 6$, $i = [1, ..., 12]$, the following Lemma tells how to infer the number of segments before a specific segment reoccurs.

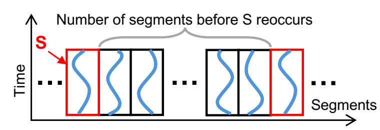

Figure 9: An illustration of UVH.

Lemma 1. *Let $\mathbf{x}$ be a UTS with a perfect period L, i.e., $\mathbf{x}_t = \mathbf{x}_{t+L}$. If $\mathbf{x}$ is divided into length-$\frac{i}{k}L$ segments, where $i, k \in \mathbb{N}^+$, the smallest number of segments, n, before any segment reoccurs, i.e., $\mathbf{x}_t = \mathbf{x}_{t+n\cdot(i/k)L}$, is given by $n = \frac{k}{GCD(i,k)}$, where GCD is the greatest common divisor.*

The proof of Lemma 1 is in Appendix C. Lemma 1 states we can calculate n given i and k. To validate the correlation between n and the difficulty of TSF, we calculate n in Fig. 8, and normalize it to range $[0, 1]$. n is small when $\frac{i}{k} = 1, 2 \rightarrow n = 1$ or $\frac{i}{k} = \frac{1}{2}, \frac{3}{2} \rightarrow n = 2$, leading to an "M"-shape curve (green). Its coincidence with the MSEs on ETTh1 and ETTm1 datasets validates our estimation of TSF difficulty, implying MAE "combines past" to forecast future. In contrast, the MSEs on Weather and Illness datasets align less with the n-values, likely due to their weaker periodic patterns.

[RQ9] Can LVMs make effective use of look-back windows? Ideally, longer look-back windows facilitate forecasting [55]. We assess MAE with different look-back window lengths in Fig. 10. The Illness dataset is excluded due to its short time series (966 time steps in total). From Fig. 10, MAE's performance improves up to a window length of 1000, after which it plateaus or declines. This may result from image transformation. Fixed-size input image in pre-trained LVMs has a pixel limit and may constrain the information captured from longer time series. Excessively long time series may distort the pixel values as they are uniformly compressed to the limited number of pixels, leading to loss of temporal information. Fortunately, contemporary LVMs handle sufficiently long windows well (1000 is long enough in many cases). Future models may extend this capability further.

5 Conclusion

In this work, we explore the potential of LVMs for time series analysis in both high-level (classification) and low-level (forecasting) tasks. By experiments with various LVMs and ablations, we offer insights into whether and how image-pretrained LVMs benefit time series tasks, hopefully helping ease their adoption across research and practical applications. Our forecasting-specific analysis highlights key limitations of current LVM forecasters, underscoring the need for improving encoder utilization, addressing inductive bias, handling longer look-back windows, and diversifying benchmarks. We hope this study complements existing research and lays the groundwork for multi-modal, agentic time series analysis.

Acknowledgments

We would like to express our sincere gratitude to Dr. Eamonn Keogh for his insightful comments on the experimental design, which helped improve our work.

References

[1] A. Ardid, D. Dempsey, C. Caudron, S. Cronin, B. Kennedy, T. Girona, D. Roman, C. Miller, S. Potter, O. D. Lamb, et al. Ergodic seismic precursors and transfer learning for short term eruption forecasting at data scarce volcanoes. *Nat. Commun.*, 16(1):1758, 2025.

[2] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

[3] H. Bao, L. Dong, S. Piao, et al. Beit: Bert pre-training of image transformers. In *ICLR*, 2022.

[4] J. O. Caro, A. H. de Oliveira Fonseca, S. A. Rizvi, M. Rosati, C. Averill, J. L. Cross, P. Mittal, E. Zappala, R. M. Dhodapkar, C. Abdallah, et al. Brainlm: A foundation model for brain activity recordings. In *ICLR*, 2024.

[5] M. Chen, L. Shen, Z. Li, X. J. Wang, J. Sun, and C. Liu. VisionTS: Visual masked autoencoders are free-lunch zero-shot time series forecasters. In *ICML*, 2025.

[6] T. Chen and C. Guestrin. XGboost: A scalable tree boosting system. In *SIGKDD*, 2016.

[7] A. Dempster, F. Petitjean, and G. I. Webb. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Min. Knowl.*, 34(5):1454–1495, 2020.

[8] Z. Dong, R. Li, Y. Wu, T. T. Nguyen, J. Chong, F. Ji, N. Tong, C. Chen, and J. H. Zhou. Brain-JEPA: Brain dynamics foundation model with gradient positioning and spatiotemporal masking. In *NeurIPS*, 2024.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[10] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi. Unsupervised scalable representation learning for multivariate time series. In *NeurIPS*, 2019.

[11] Y. Gong, Y.-A. Chung, and J. Glass. AST: Audio spectrogram transformer. *Interspeech*, 2021.

[12] Y. Gong, C.-I. Lai, Y.-A. Chung, and J. Glass. SSAST: Self-supervised audio spectrogram transformer. In *AAAI*, 2022.

[13] A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[14] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao. ADBench: Anomaly detection benchmark. In *NeurIPS*, 2022.

[15] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[16] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, and H. Li. FlowFormer: A transformer architecture for optical flow. In *ECCV*, 2022.

[17] Y. Jiang, K. Ning, Z. Pan, X. Shen, J. Ni, W. Yu, A. Schneider, H. Chen, Y. Nevmyvaka, and D. Song. Multi-modal time series analysis: A tutorial and survey. *arXiv preprint arXiv:2503.13709*, 2025.

[18] Y. Jiang, Z. Pan, X. Zhang, S. Garg, A. Schneider, Y. Nevmyvaka, and D. Song. Empowering time series analysis with large language models: a survey. In *IJCAI*, 2024.

[19] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, et al. Time-LLM: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[20] M. Jin, Y. Zhang, W. Chen, K. Zhang, Y. Liang, B. Yang, J. Wang, S. Pan, and Q. Wen. Position: What can large language models tell us about time series analysis. In *ICML*, 2024.

[21] H. Karami, M.-A. Hartley, D. Atienza, and A. Ionescu. TimEHR: Image-based time series generation for electronic health records. *arXiv preprint arXiv:2402.06318*, 2024.

[22] W. Kim, B. Son, and I. Kim. ViLT: Vision-and-language transformer without convolution or region supervision. In *ICML*, 2021.

[23] N. Kitaev, Ł. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[24] I. Koprinska, D. Wu, and Z. Wang. Convolutional neural networks for energy time series forecasting. In *IJCNN*, 2018.

[25] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*, 2018.

[26] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, 2015.

[27] X. Li, Y. Kang, and F. Li. Forecasting with time series imaging. *Expert Syst. Appl.*, 160:113680, 2020.

[28] X. Li, V. Metsis, H. Wang, and A. H. H. Ngu. TTS-GAN: A transformer-based time-series generative adversarial network. In *AIME*, 2022.

[29] Z. Li, S. Li, and X. Yan. Time series as images: Vision transformer for irregularly sampled time series. In *NeurIPS*, 2023.

[30] S. Lin, W. Lin, W. Wu, H. Chen, and J. Yang. SparseTSF: Modeling long-term time series forecasting with 1k parameters. In *ICML*, 2024.

[31] H. Liu, H. Kamarthi, Z. Zhao, S. Xu, S. Wang, Q. Wen, T. Hartvigsen, F. Wang, and B. A. Prakash. How can time series analysis benefit from multiple modalities? A survey and outlook. *arXiv preprint arXiv:2503.11835*, 2025.

[32] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[33] P. Liu, H. Guo, T. Dai, N. Li, J. Bao, X. Ren, Y. Jiang, and S.-T. Xia. CALF: Aligning LLMs for time series forecasting via cross-modal fine-tuning. In *AAAI*, 2025.

[34] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *ICLR*, 2022.

[35] Y. Liu, H. Wu, J. Wang, and M. Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *NeurIPS*, 2022.

[36] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[37] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4):818, 2017.

[38] M. A. Morid, O. R. L. Sheng, and J. Dunbar. Time series prediction using deep learning methods in healthcare. *ACM TMIS*, 14(1):1–29, 2023.

[39] N. Namura, Y. Ichikawa, et al. Training-free time-series anomaly detection: Leveraging image foundation models. *arXiv preprint arXiv:2408.14756*, 2024.

[40] J. Ni, Z. Zhao, C. Shen, H. Tong, D. Song, W. Cheng, D. Luo, and H. Chen. Harnessing vision models for time series analysis: A survey. In *IJCAI*, 2025.

[41] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *ICLR*, 2023.

[42] V. Prithyani, M. Mohammed, R. Gadgil, R. Buitrago, V. Jain, and A. Chadha. On the feasibility of vision-language models for time-series classification. *arXiv:2412.17304*, 2024.

[43] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[44] M. Tan, M. Merrill, V. Gupta, T. Althoff, and T. Hartvigsen. Are language models actually useful for time series forecasting? In *NeurIPS*, 2024.

[45] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

[46] A. Trindade. Electricity Load Diagrams. UCI Machine Learning Repository, 2015.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[48] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey. In *IJCAI*, 2023.

[49] C. Wimmer and N. Rekabsaz. Leveraging vision-language models for granular market change prediction. *arXiv:2301.10166*, 2023.

[50] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi. ETSformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.

[51] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. TimesNet: Temporal 2d-variation modeling for general time series analysis. In *ICLR*, 2023.

[52] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, 2021.

[53] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu. SimMIM: A simple framework for masked image modeling. In *CVPR*, 2022.

[54] L. Yang, Y. Wang, X. Fan, I. Cohen, J. Chen, Y. Zhao, and Z. Zhang. ViTime: A visual intelligence-based foundation model for time series forecasting. *arXiv preprint arXiv:2407.07311*, 2024.

[55] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? In *AAAI*, 2023.

[56] Z. Zeng, R. Kaur, S. Siddagangappa, et al. From pixels to predictions: Spectrogram and vision transformer for better time series forecasting. In *ICAIF*, 2023.

[57] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff. A transformer-based framework for multivariate time series representation learning. In *SIGKDD*, 2021.

[58] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *AAAI*, 2019.

[59] T. Zhang, Y. Zhang, W. Cao, J. Bian, X. Yi, S. Zheng, and J. Li. Less is more: Fast multivariate time series forecasting with light sampling-oriented MLP structures. arxiv. *arXiv preprint arXiv:2207.01186*, 2022.

[60] X. Zhang, R. R. Chowdhury, R. K. Gupta, and J. Shang. Large language models for time series: A survey. In *IJCAI*, 2024.

[61] S. Zhong, W. Ruan, M. Jin, H. Li, Q. Wen, and Y. Liang. Time-VLM: Exploring multimodal vision-language models for augmented time series forecasting. *arXiv preprint arXiv:2502.04395*, 2025.

[62] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, 2021.

[63] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, 2022.

[64] T. Zhou, P. Niu, L. Sun, R. Jin, et al. One fits all: Power general time series analysis by pretrained LM. In *NeurIPS*, 2023.

[65] Z. Zhou and R. Yu. Can LLMs understand time series anomalies? In *ICLR*, 2025.

[66] J. Zhuang, L. Yan, Z. Zhang, R. Wang, J. Zhang, and Y. Gu. See it, think it, sorted: Large multi-modal models are few-shot time series anomaly analyzers. *arXiv preprint arXiv:2411.02465*, 2024.

A Experimental Setup

A.1 Benchmarks

Time Series Classification. For TSC, following [51, 64], our experiments are conducted on 10 multivariate benchmark datasets from UEA archive [2], which span diverse domains, including chemical analysis, cognitive neuroscience, gesture recognition, biomedical signal processing, speech recognition and traffic analysis. Table 6 summarizes the statistics of the datasets.

Dataset	Variates	Series Length	Dataset Size	Classes
EthanolConcentration	3	1751	(261, 263, 263)	4
FaceDetection	144	62	(5890, 3524, 3524)	2
Handwriting	3	152	(150, 850, 850)	26
Heartbeat	61	405	(204, 205,205)	2
Japanese Vowels	12	29	(270, 370, 370)	9
PEMS-SF	963	144	(267, 173, 173)	7
SelfRegulationSCP1	6	896	(268, 293, 293)	2
SelfRegulationSCP2	7	1152	(200, 180, 180)	2
SpokenArabicDigits	13	93	(6599, 2199, 2199)	10
UWaveGestureLibrary	3	315	(120, 320, 320)	8

Table 6: Statistics of the datasets for TSC. “Dataset Size” is organized in (Train, Validation, Test).

Time Series Forecasting. For TSF, following [62, 52, 41, 55, 44, 5], our experiments are conducted on 8 widely used benchmark datasets. 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 Illness dataset keeps weekly counts of patients and the influenza-like illness ratio from the United States. The Traffic dataset measures hourly road occupancy rates from sensors on San Francisco freeways. The Electricity dataset records hourly electricity consumption of Portuguese clients. Table 7 summarizes the statistics of the datasets.

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
Illness	7	966	(617, 74, 170)	Weekly
Traffic	862	17544	(12185, 1757, 3509)	Hourly
Electricity	21	26304	(18317, 2633, 5261)	Hourly

Table 7: Statistics of the datasets for TSF. “Dataset Size” is organized in (Train, Validation, Test).

A.2 Baselines

For TSC, following [64], 18 conventional and SOTA baselines are included. For TSF, following [41, 44, 5], 8 representative LLM-based, Transfomer-based, and non-Transformer-based baselines are included. Since several baselines are used in both TSC and TSF tasks (*e.g.*, GPT4TS, Autoformer, Dlinear, *etc.*), there are 21 distinct baselines, which are described as follows.

- GPT4TS [64] is a foundation model built on GPT for various of time series tasks.
- Time-LLM [19] implements reprogramming to align time series with language so as to leverage pre-trained LLMs.
- CALF [33] is built upon LLMs by designing a cross attention and feature regularization loss to align time series with language.
- PatchTST [41] divides time series into subsequence-based patches, which is then modeled as tokens through Transformer encoders with channel independence strategy.

- Flowformer [16] introduces a linear-time attention mechanism named Flow-Attention without using specific inductive biases for time series forecasting.
- Informer [62] is a Transformer-based model that designs a ProbSparse attention mechanism to reduce time complexity on long time series.
- Transformer [47] is the most traditional encoder-decoder structure which can model time series with attention mechanism.
- Stationary [35] combines series stationarization and de-stationary attention to solve the over-stationarization problem in time series forecasting.
- Refromer [23] applies locality-sensitive hashing and reversible residual layers to improve the efficiency of using Transformers to model long time series.
- Autoformer [52] replaces the attention block of Transformer with the Auto-Correlation mechanism which can enhance both efficiency and accuracy.
- ETSformer [50] decomposes an input time series into interpretable components with exponential smoothing attention and frequency attention for time series forecasting.
- Pyraformer [34] designs a pyramidal attention module with inter-scale tree structures and intra-scale neighboring connections to capture multi-resolution temporal dependencies.
- FEDformer [63] combines seasonal-trend decomposition with a frequency-enhanced Transformer to capture both global patterns and detailed structures in time series.
- Rocket [7] achieves accurate time series classification by using linear classifiers with random convolutional kernels.
- XGBoost [6] is an efficient implementation of gradient boost decision trees for both classification and regression tasks.
- Dlinear [55] is a linear model that decomposes an input time series into seasonal component and trend component, and then models them with linear layers.
- LightTS [59] is an efficient MLP-based architecture for multivariate time series forecasting by leveraging interval and continuous down-sampling to preserve temporal patterns.
- TimesNet [51] transforms time series into a 2D image-like representation using period-based patching, and then models the transformed time series with inception blocks.
- TCN [10] is a type of convolutional neural network that use causal, dilated convolutions with residual connections to model the temporal dependencies in time series.
- LSTNet [25] integrates RNNs and CNNs to capture temporal patterns in time series.
- LSSL [13] is proposed based on a new parameterization for state space model to capture the long-term dependencies in time series.

A.3 Evaluation Metrics

For TSC, following [51, 64], accuracy (in percentage) is used as the evaluation metric. For TSF, following [41, 55, 44, 5], Mean Squared Error (MSE) and Mean Absolute Error (MAE) are used as the evaluation metrics. Eq (1) defines MSE and MAE.

$$\textbf{MSE} = \frac{1}{D\cdot T}\sum_{d=1}^{D}\sum_{t=1}^{T}\|\hat{\mathbf{Y}}_{dt}-\mathbf{Y}_{dt}\|_2^2, \qquad \textbf{MAE} = \frac{1}{D\cdot T}\sum_{d=1}^{D}\sum_{t=1}^{T}\|\hat{\mathbf{Y}}_{dt}-\mathbf{Y}_{dt}\|_1 \tag{1}$$

where $\hat{\mathbf{Y}} \in \mathbb{R}^{D\times T}$ stands for the prediction at T future time steps of D variates, $\mathbf{Y}$ stands for the ground truth, $\|\cdot\|_2$ is ℓ_2 norm, and $\|\cdot\|_1$ is ℓ_1 norm.

Following [41, 55, 44], for fair comparison, we adopt the standard evaluation protocol. In particular, the look-back window length is set to $H = 336$. The prediction lengths is set to $T \in \{96, 192, 336, 720\}$ for all datasets except for Illness dataset. For Illness dataset, because of its limited total length of 966 time steps, shorter look-back window of $H = 104$ and prediction lengths $T \in \{24, 36, 48, 60\}$ are employed by following [41, 55, 44]. Unless otherwise noted, this configuration is applied to all of the experiments on TSF.

A.4 Implementation Details

As described in §4.1, 4 pre-trained LVMs have been included in our experiments. For ViT and Swin, we use the checkpoints `ViT_B_16_Weights.IMAGENET1K_V1` and `Swin_B_Weights.IMAGENET1K_V1` respectively from *PyTorch*, which are pre-trained on $224 \times 224 \times 3$ sized images. For MAE, we use the checkpoint released by *Meta Research*[1], which is pre-trained on $224 \times 224 \times 3$ sized images with ViT-Base backbone. For SimMIM, we use the checkpoint released by *Microsoft*[2], which is pre-trained on $192 \times 192 \times 3$ sized images with Swin-Base backbone.

For TSC task, we fine-tune the LVMs using Adam optimizer with learning rate 0.0001 and batch size 32. The training runs up to a maximum of 30 epochs on the training set. Early stopping is applied after 8 consecutive epochs of no improvement is observed on the validation set.

For TSF task, we use Adam optimizer with learning rate 0.0001. For ETT and Illness datasets, the batch size is set to 32. For Weather, Traffic and Electricity datasets, the batch size is set to 256. The training runs up to 20 epochs on the training set. Early stopping is applied after 3 consecutive epochs of no improvement is observed on the validation set.

All experiments are repeated three times, and the final result is obtained by taking the average. Unless otherwise noted, the above training configuration is applied to all experiments.

The experiments are conducted on NVIDIA RTX 6000 Ada Generation GPUs with 48GB memory. All implementations are based on PyTorch 2.6.0 and utilize CUDA 12.4 for training.

A.5 Imaging Methods

In this section, we elaborate Gramian Angular Field (GAF) and Univariate Heatmap (UVH), as they are the most frequently used imaging methods in our experiments. For more details about GAF, UVH, and other imaging methods, we refer readers to [40].

Gramian Angular Field (GAF). Given a univariate time series $\mathbf{x} = [x_1, ..., x_T] \in \mathbb{R}^{1\times T}$, where x_i ($1 \le i \le T$) is the value at time step i, GAF applies Min-Max scaling to normalize each x_i to $\hat{x}_i \in [0, 1]$. This normalization allows each time step to be mapped into polar coordinates with angular component $\phi_i = \arccos(\hat{x}_i)$ and radial component $r_i = i/N$, where N is a constant factor.

In Gramian Sum Angular Field (GSAF), the (i, j)-th entry encodes the temporal correlation between time steps i and j, which is computed as $\cos(\phi_i + \phi_j)$ and can be further expanded as following.

$$cos(\phi_i + \phi_j) = \hat{x}_i\hat{x}_j - \sqrt{1-\hat{x}_i^2}\sqrt{1-\hat{x}_j^2} \tag{2}$$

The resulting GAF is a matrix of size $T \times T$, with (i, j)-th entry defined as $\cos(\phi_i + \phi_j)$, which captures the pairwise temporal correlations among all time steps. For a multivariate time series $\mathbf{X} \in \mathbb{R}^{d\times T}$, the resulting GAF consists of d individual $T \times T$ matrices.

Univariate Heatmap (UVH). Given a univariate time series $\mathbf{x} \in \mathbb{R}^{1\times T}$, UVH applies Fast Fourier Transform (FFT) to compute the Fourier coefficient of each frequency component f_i, where $f_i \in [1, \lfloor T/2 \rfloor]$. Then it identifies the dominant frequency f_L with the largest coefficient amplitude, and sets the potential period length as $L = \lceil T/f_L \rceil$. Next, $\mathbf{x}$ is left-padded to a length-$\hat{T}$ time series $\hat{\mathbf{x}}$, where $\hat{T}$ is a multiple of L. The padded time series $\hat{\mathbf{x}}$ is subsequently reshaped into a 2D image of size $L \times \hat{T}/L$ by stacking it subsequences of length L.

Segment length selection for UVH. To identify the best segment length for UVH, FFT is applied on a long look-back window of 1152 time steps on all datasets except for Illness dataset, where 104 time steps is used to accommodate its short time series. Table 8 summarizes the top-3 potential periods with the highest Fourier coefficients on each TSF dataset, along with the segment length L used in the subsequent experiments involving UVH imaging method.

[1] https://github.com/facebookresearch/mae

[2] https://github.com/microsoft/SimMIM

	ETTh1, ETTh2	ETTm1, ETTm2	Weather	Illness	Traffic	Electricity
Top 3 Period	{24, 576, 384}	{96, 576, 384}	{144, 72, 576}	{52, 26, 17}	{24, 12, 168}	{24, 164, 82}
Segment Length L	24	96	144	52	24	24

Table 8: Top-3 potential periods by FFT and segment lengths for UVH on 8 TSF datasets.

B Full Experimental Results

B.1 Full Results of Time Series Classification

Table 9 provides the full results of the compared methods on 10 benchmark datasets for TSC. The LVM results are averaged over 3 runs. The corresponding standard deviations reported in Table 10.

Dataset	**MAE**	**ViT**	**XGBoost**	**Rocket**	**LSTNet**	**LSSL**	**TCN**	**Trans.**	**Re.**	**In.**	**Pyra.**	**Auto.**	**Station.**	**FED.**	**ETS.**	**Flow.**	**Dlinear**	**LightTS**	**TimesNet**	**GPT4TS**
EthanolConcentration	41.4	49.4	43.7	45.2	39.9	31.1	28.9	32.7	31.9	31.6	30.8	31.6	32.7	31.2	28.1	33.8	32.6	29.7	35.7	34.2
FaceDetection	65.4	67.4	63.3	64.7	65.7	66.7	52.8	67.3	68.6	67.0	65.7	68.4	68.0	66.0	66.3	67.6	68.0	67.5	68.6	69.2
Handwriting	39.5	36.4	15.8	58.8	25.8	24.6	53.3	32.0	27.4	32.8	29.4	36.7	31.6	28.0	32.5	33.8	27.0	26.1	32.1	32.7
Heartbeat	86.8	74.6	73.2	75.6	77.1	72.7	75.6	76.1	77.1	80.5	75.6	74.6	73.7	73.7	71.2	77.6	75.1	75.1	78.0	77.2
Japanese Vowels	95.4	98.3	86.5	96.2	98.1	98.4	98.9	98.7	97.8	98.9	98.4	96.2	99.2	98.4	95.9	98.9	96.2	96.2	98.4	98.6
PEMS-SF	84.4	84.2	98.3	75.1	86.7	86.1	68.8	82.1	82.7	81.5	83.2	82.7	87.3	80.9	86.0	83.8	75.1	88.4	89.6	87.9
SelfRegulationSCP1	95.2	97.2	84.6	90.8	84.0	90.8	84.6	92.2	90.4	90.1	88.1	84.0	89.4	88.7	89.6	92.5	87.3	89.8	91.8	93.2
SelfRegulationSCP2	59.4	58.8	48.9	53.3	52.8	52.2	55.6	53.9	56.7	53.3	53.3	50.6	57.2	54.4	55.0	56.1	50.5	51.1	57.2	59.4
SpokenArabicDigits	98.5	98.5	69.6	71.2	100.0	100.0	95.6	98.4	97.0	100.0	99.6	100.0	100.0	100.0	100.0	98.8	81.4	100.0	99.0	99.2
UWaveGestureLibrary	85.0	88.4	75.9	94.4	87.8	85.9	88.4	85.6	85.6	85.6	83.4	85.9	87.5	85.3	85.0	86.6	82.1	80.3	85.3	88.1
Average	75.1	**75.3**	66.0	72.5	71.8	70.9	70.3	71.9	71.5	72.1	70.8	71.1	72.7	70.7	71.0	73.0	67.5	70.4	73.6	74.0
# Wins	2	**3**	1	1	1	1	1	0	0	1	0	1	2	1	1	0	0	1	0	2

Table 9: Accuracy (%) of the compared methods in TSC on 10 benchmark datasets. Red numbers are the the best results. # Wins is the number of times the method performs the best.

Dataset	**MAE**	**ViT**
EthanolConcentration	41.4 ± 0.5	49.4 ± 0.9
FaceDetection	65.4 ± 1.2	67.4 ± 1.5
Handwriting	39.5 ± 1.5	36.4 ± 1.3
Heartbeat	86.8 ± 2.1	74.6 ± 0.6
Japanese Vowels	95.4 ± 0.3	98.3 ± 0.3
PEMS-SF	84.4 ± 0.4	84.2 ± 0.5
SelfRegulationSCP1	95.2 ± 0.6	97.2 ± 0.9
SelfRegulationSCP2	59.4 ± 1.5	58.8 ± 1.3
SpokenArabicDigits	98.5 ± 0.5	98.5 ± 0.5
UWaveGestureLibrary	85.0 ± 1.7	88.4 ± 1.4

Table 10: Standard deviation of LVMs on TSC datasets.

B.2 Full Results of Time Series Forecasting

Table 11 provides the full result of the compared methods on 8 benchmark datasets for TSF. The results of LVMs are averaged over 3 runs with standard deviations reported in Table 12.

Method		MAE		ViT		Time-LLM		GPT4TS		CALF		Dlinear		PatchTST		TimesNet		FEDformer		Autoformer	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.356	0.383	0.398	0.401	0.376	0.402	0.370	0.389	0.370	0.393	0.375	0.399	0.370	0.399	0.384	0.402	0.376	0.419	0.449	0.459
	192	0.395	0.406	0.439	0.445	0.407	0.421	0.412	0.413	0.429	0.426	0.405	0.416	0.413	0.421	0.436	0.429	0.420	0.448	0.500	0.482
	336	0.417	0.424	0.462	0.458	0.430	0.438	0.448	0.431	0.451	0.440	0.439	0.443	0.422	0.436	0.491	0.469	0.459	0.465	0.521	0.496
	720	0.467	0.463	0.479	0.491	0.457	0.468	0.441	0.449	0.476	0.466	0.472	0.490	0.447	0.466	0.521	0.500	0.506	0.507	0.514	0.512
ETTh2	96	0.297	0.341	0.302	0.355	0.286	0.346	0.280	0.335	0.284	0.336	0.289	0.353	0.274	0.336	0.340	0.374	0.358	0.397	0.346	0.388
	192	0.356	0.386	0.394	0.411	0.361	0.391	0.348	0.380	0.353	0.378	0.383	0.418	0.339	0.379	0.402	0.414	0.429	0.439	0.456	0.452
	336	0.371	0.402	0.423	0.429	0.390	0.414	0.380	0.405	0.361	0.394	0.448	0.465	0.329	0.380	0.452	0.452	0.496	0.487	0.482	0.486
	720	0.403	0.430	0.438	0.449	0.405	0.434	0.406	0.436	0.406	0.428	0.605	0.551	0.379	0.422	0.462	0.468	0.463	0.474	0.515	0.511
ETTm1	96	0.284	0.333	0.344	0.384	0.291	0.341	0.300	0.340	0.323	0.350	0.299	0.343	0.290	0.342	0.338	0.375	0.379	0.419	0.505	0.475
	192	0.328	0.363	0.414	0.425	0.341	0.369	0.343	0.368	0.375	0.376	0.335	0.365	0.332	0.369	0.374	0.387	0.426	0.441	0.553	0.496
	336	0.357	0.384	0.411	0.427	0.359	0.379	0.376	0.386	0.411	0.401	0.369	0.386	0.366	0.392	0.410	0.411	0.445	0.459	0.621	0.537
	720	0.411	0.417	0.466	0.451	0.433	0.419	0.431	0.416	0.476	0.438	0.425	0.421	0.416	0.420	0.478	0.450	0.543	0.490	0.671	0.561
ETTm2	96	0.173	0.258	0.179	0.265	0.162	0.248	0.163	0.249	0.177	0.255	0.167	0.269	0.165	0.255	0.187	0.267	0.203	0.287	0.255	0.339
	192	0.231	0.297	0.262	0.319	0.235	0.304	0.222	0.291	0.245	0.300	0.224	0.303	0.220	0.292	0.249	0.309	0.269	0.328	0.281	0.340
	336	0.282	0.340	0.346	0.371	0.280	0.329	0.273	0.327	0.309	0.341	0.281	0.342	0.274	0.329	0.321	0.351	0.325	0.366	0.339	0.372
	720	0.386	0.413	0.411	0.392	0.366	0.382	0.357	0.376	0.402	0.395	0.397	0.421	0.362	0.385	0.408	0.403	0.421	0.415	0.433	0.432
Weather	96	0.146	0.191	0.162	0.219	0.155	0.199	0.148	0.188	0.168	0.207	0.176	0.237	0.149	0.198	0.172	0.220	0.217	0.296	0.266	0.336
	192	0.194	0.238	0.196	0.244	0.223	0.261	0.192	0.230	0.216	0.251	0.220	0.282	0.194	0.241	0.219	0.261	0.276	0.336	0.307	0.367
	336	0.243	0.275	0.250	0.286	0.251	0.279	0.246	0.273	0.271	0.292	0.265	0.319	0.245	0.282	0.280	0.306	0.339	0.380	0.359	0.395
	720	0.318	0.328	0.329	0.342	0.345	0.342	0.320	0.328	0.350	0.345	0.333	0.362	0.314	0.334	0.365	0.359	0.403	0.428	0.419	0.428
Illness	24	1.977	0.921	1.989	0.941	1.792	0.807	1.869	0.823	1.460	0.788	2.215	1.081	1.319	0.754	2.317	0.934	3.228	1.260	3.483	1.287
	36	1.812	0.872	2.123	1.002	1.833	0.833	1.853	0.854	1.573	0.837	1.963	0.963	1.430	0.834	1.972	0.920	2.679	1.080	3.103	1.148
	48	1.743	0.856	2.200	1.032	2.269	1.012	1.886	0.855	1.784	0.890	2.130	1.024	1.553	0.815	2.238	0.940	2.622	1.078	2.669	1.085
	60	1.816	0.881	2.404	1.087	2.177	0.925	1.877	0.877	1.982	0.962	2.368	1.096	1.470	0.788	2.027	0.928	2.857	1.157	2.770	1.125
Traffic	96	0.346	0.232	0.403	0.330	0.392	0.267	0.396	0.264	0.416	0.274	0.410	0.282	0.360	0.249	0.593	0.321	0.587	0.366	0.613	0.388
	192	0.376	0.245	0.411	0.334	0.409	0.271	0.412	0.268	0.430	0.276	0.423	0.287	0.379	0.256	0.617	0.336	0.604	0.373	0.616	0.382
	336	0.389	0.252	0.429	0.335	0.434	0.296	0.421	0.273	0.451	0.286	0.436	0.296	0.392	0.264	0.629	0.336	0.621	0.383	0.622	0.337
	720	0.432	0.293	0.477	0.371	0.451	0.291	0.455	0.291	0.478	0.301	0.466	0.315	0.432	0.286	0.640	0.350	0.626	0.382	0.660	0.408
Electricity	96	0.127	0.217	0.152	0.244	0.137	0.233	0.141	0.239	0.147	0.240	0.140	0.237	0.129	0.222	0.168	0.272	0.193	0.308	0.201	0.317
	192	0.148	0.237	0.164	0.249	0.152	0.247	0.158	0.253	0.163	0.254	0.153	0.249	0.157	0.240	0.184	0.289	0.201	0.315	0.222	0.334
	336	0.163	0.253	0.173	0.275	0.169	0.267	0.172	0.266	0.178	0.270	0.169	0.267	0.163	0.259	0.198	0.300	0.214	0.329	0.231	0.338
	720	0.199	0.293	0.202	0.294	0.200	0.290	0.207	0.293	0.215	0.300	0.203	0.301	0.197	0.290	0.220	0.320	0.246	0.355	0.254	0.361
# Wins		28		0		5		14		1		0		20		0		0		0	

Table 11: MSE and MAE evaluation of the compared methods in TSF on benchmark datasets. Red (Blue) numbers are the best (second best) results on each prediction length per dataset. # Wins is the number of times the method performs the best.

Method		MAE		ViT	
Metrics		MSE	MAE	MSE	MAE
ETTh1	96	0.356 ± 0.001	0.383 ± 0.005	0.398 ± 0.011	0.401 ± 0.012
	192	0.395 ± 0.001	0.406 ± 0.001	0.439 ± 0.005	0.445 ± 0.003
	336	0.417 ± 0.001	0.424 ± 0.001	0.462 ± 0.004	0.458 ± 0.004
	720	0.467 ± 0.012	0.463 ± 0.010	0.479 ± 0.011	0.491 ± 0.008
ETTh2	96	0.297 ± 0.000	0.341 ± 0.004	0.302 ± 0.001	0.355 ± 0.000
	192	0.356 ± 0.005	0.386 ± 0.011	0.394 ± 0.001	0.411 ± 0.001
	336	0.371 ± 0.003	0.402 ± 0.004	0.423 ± 0.003	0.429 ± 0.001
	720	0.403 ± 0.001	0.430 ± 0.005	0.438 ± 0.005	0.449 ± 0.002
ETTm1	96	0.284 ± 0.003	0.333 ± 0.004	0.344 ± 0.001	0.384 ± 0.002
	192	0.328 ± 0.001	0.363 ± 0.002	0.414 ± 0.003	0.425 ± 0.003
	336	0.357 ± 0.001	0.384 ± 0.001	0.411 ± 0.002	0.427 ± 0.007
	720	0.411 ± 0.002	0.417 ± 0.001	0.466 ± 0.003	0.451 ± 0.002
ETTm2	96	0.173 ± 0.005	0.258 ± 0.004	0.179 ± 0.003	0.265 ± 0.004
	192	0.231 ± 0.004	0.297 ± 0.003	0.262 ± 0.002	0.319 ± 0.001
	336	0.282 ± 0.001	0.340 ± 0.004	0.346 ± 0.001	0.371 ± 0.003
	720	0.386 ± 0.002	0.413 ± 0.003	0.411 ± 0.002	0.392 ± 0.004
Weather	96	0.146 ± 0.000	0.191 ± 0.002	0.162 ± 0.001	0.219 ± 0.003
	192	0.194 ± 0.001	0.238 ± 0.002	0.196 ± 0.002	0.244 ± 0.003
	336	0.243 ± 0.000	0.275 ± 0.001	0.250 ± 0.001	0.286 ± 0.000
	720	0.318 ± 0.001	0.328 ± 0.001	0.329 ± 0.002	0.342 ± 0.002
Illness	24	1.977 ± 0.017	0.921 ± 0.003	1.989 ± 0.011	0.941 ± 0.004
	36	1.812 ± 0.014	0.872 ± 0.009	2.123 ± 0.006	1.002 ± 0.003
	48	1.743 ± 0.029	0.856 ± 0.012	2.200 ± 0.009	1.032 ± 0.005
	60	1.816 ± 0.022	0.881 ± 0.008	2.404 ± 0.018	1.087 ± 0.011
Traffic	96	0.346 ± 0.004	0.232 ± 0.003	0.403 ± 0.003	0.330 ± 0.002
	192	0.376 ± 0.006	0.245 ± 0.002	0.411 ± 0.001	0.334 ± 0.000
	336	0.389 ± 0.004	0.252 ± 0.003	0.429 ± 0.002	0.335 ± 0.005
	720	0.432 ± 0.002	0.293 ± 0.005	0.477 ± 0.004	0.371 ± 0.002
Electricity	96	0.127 ± 0.001	0.217 ± 0.000	0.152 ± 0.001	0.244 ± 0.001
	192	0.148 ± 0.004	0.237 ± 0.000	0.164 ± 0.003	0.249 ± 0.001
	336	0.163 ± 0.001	0.253 ± 0.002	0.173 ± 0.002	0.275 ± 0.003
	720	0.199 ± 0.002	0.293 ± 0.001	0.202 ± 0.001	0.294 ± 0.003

Table 12: Standard deviation of LVMs on TSF datasets.

B.3 Full Results of RQ1: What type of LVM best fits TSC (TSF) task?

The detailed performance comparison between self-supervised LVMs and supervised LVMs using the best imaging method on TSC (*i.e.*, GAF) and TSF (*i.e.*, UVH) tasks are provided in Table 13 and Table 14, respectively. For TSC, supervised and self-supervised LVMs perform comparably, while for TSF, self-supervised LVMs outperform their supervised counterparts.

Dataset	MAE	SimMiM	ViT	Swin
UWaveGestureLibrary	85.0	83.1	88.4	78.9
SpokenArabicDigits	98.5	88.2	98.5	87.3
Handwriting	39.5	29.8	36.4	33.1
FaceDetection	65.4	57.8	67.4	50.3
Average	72.1	64.7	72.6	62.4

Table 13: Accuracy (%) comparison between self-supervised LVMs and supervised LVMs on TSC benchmark datasets. Red numbers indicate the best performance for each dataset.

Dataset	**Model**	**Self-Supervised** MAE		**Self-Supervised** SimMIM		**Supervised** ViT		**Supervised** Swin	
	Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.356	0.383	0.362	0.383	0.398	0.401	0.407	0.429
	192	0.395	0.406	0.407	0.412	0.439	0.445	0.442	0.458
	336	0.417	0.424	0.422	0.417	0.462	0.458	0.467	0.481
	720	0.467	0.463	0.462	0.455	0.479	0.491	0.470	0.497
	Average	0.409	0.419	0.413	0.417	0.445	0.449	0.447	0.466
ETTm1	96	0.284	0.333	0.311	0.350	0.344	0.384	0.308	0.360
	192	0.328	0.363	0.335	0.367	0.414	0.425	0.350	0.381
	336	0.357	0.384	0.356	0.382	0.411	0.427	0.385	0.407
	720	0.411	0.417	0.400	0.413	0.466	0.451	0.430	0.437
	Average	0.345	0.374	0.351	0.378	0.409	0.422	0.368	0.396
Weather	96	0.146	0.191	0.148	0.196	0.162	0.219	0.163	0.216
	192	0.194	0.238	0.196	0.243	0.196	0.244	0.214	0.262
	336	0.243	0.275	0.244	0.276	0.250	0.286	0.270	0.298
	720	0.318	0.328	0.340	0.340	0.329	0.342	0.345	0.348
	Average	0.225	0.258	0.232	0.264	0.234	0.273	0.248	0.281
Illness	24	1.977	0.921	1.934	0.902	1.989	0.941	1.990	0.942
	36	1.812	0.872	1.754	0.825	2.123	1.002	2.003	0.951
	48	1.743	0.856	1.715	0.867	2.200	1.032	2.084	0.991
	60	1.816	0.881	1.673	0.877	2.404	1.087	2.128	1.007
	Average	1.837	0.883	1.769	0.868	2.179	1.016	2.051	0.973

Table 14: MSE and MAE Comparison between self-supervised LVMs and supervised LVMs on TSF datasets. Red numbers indicate the best performance for each prediction length per dataset.

B.4 Full Results of RQ2: Which imaging method best fits TSC (TSF) task?

This section provides detailed performance comparison of 8 imaging methods, including GAF, MVH, RP, STFT, Wavelet (Wave.), Filterbank (Filter.), UVH, and Line Plot. The best LVMs for TSC (*i.e.*, ViT) and TSF (*i.e.*, MAE) are used. Table 15 and Table 16 summarize the results for TSC and TSF, respectively. Unlike TSF, in which UVH demonstrates a clear advantage on the four datasets in Table 16, for TSC, we find ranking the compared methods using critical difference diagram (Fig. 4) over all TSC benchmark datasets gives higher confidence to identify the best imaging method (*i.e.*, GAF). Thus Table 15 includes the results on all TSC benchmark datasets.

Dataset	GAF	MVH	RP	STFT	Wave.	Filter.	UVH	Lineplot
EthanolConcentration	49.4	30.7	43.7	31.9	27.3	28.1	28.5	25.2
FaceDetection	67.4	68.3	65.5	61.1	63.9	64.7	67.7	50.3
Handwriting	36.4	30.8	45.1	28.2	34.0	22.3	25.8	15.9
Heartbeat	74.6	77.5	71.7	74.7	72.6	73.1	78.0	53.7
Japanese Vowels	98.3	97.8	87.8	94.8	94.9	97.0	96.4	65.7
PEMS-SF	84.2	87.2	80.1	68.5	84.7	71.2	88.1	73.4
SelfRegulationSCP1	97.2	90.4	98.6	90.7	76.7	55.6	91.8	85.3
SelfRegulationSCP2	58.8	53.3	54.4	52.7	54.4	52.2	52.8	44.5
SpokenArabicDigits	98.5	97.5	98.4	97.9	96.1	95.0	97.0	68.1
UWaveGestureLibrary	88.4	88.7	91.8	86.2	86.3	52.1	84.3	74.0
Average	75.3	72.2	73.7	68.7	69.1	61.1	71.0	55.6

Table 15: Accuracy (%) comparison of 8 imaging methods on TSC benchmark datasets. Red numbers indicate the best performance for each dataset.

Imaging Method		GAF		MVH		RP		STFT		Wave.		Filter.		UVH		Lineplot	
Dataset	**Metrics**	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.986	0.783	0.484	0.471	0.969	0.771	0.534	0.533	0.621	0.582	0.820	0.684	0.356	0.383	0.902	0.751
	192	1.004	0.797	0.575	0.517	0.971	0.775	0.621	0.587	0.650	0.600	0.864	0.707	0.395	0.406	1.204	0.894
	336	1.038	0.820	0.623	0.546	0.989	0.788	0.602	0.573	0.681	0.616	0.827	0.693	0.417	0.424	1.223	0.901
	720	1.008	0.812	0.737	0.612	1.062	0.825	0.669	0.621	0.699	0.633	0.858	0.720	0.467	0.463	1.150	0.852
	Average	1.009	0.803	0.605	0.537	0.998	0.790	0.607	0.579	0.663	0.608	0.842	0.701	0.409	0.419	1.120	0.850
ETTm1	96	0.836	0.729	0.310	0.352	0.849	0.719	0.420	0.470	0.449	0.490	0.793	0.648	0.284	0.333	0.842	0.735
	192	0.830	0.717	0.386	0.400	0.865	0.726	0.466	0.496	0.504	0.524	0.798	0.649	0.328	0.363	0.840	0.726
	336	0.853	0.725	0.393	0.402	0.872	0.728	0.506	0.519	0.532	0.535	0.883	0.690	0.357	0.384	0.841	0.726
	720	0.865	0.726	0.488	0.467	0.928	0.754	0.543	0.536	0.586	0.563	0.899	0.703	0.411	0.417	0.872	0.741
	Average	0.846	0.724	0.394	0.405	0.879	0.732	0.484	0.505	0.518	0.528	0.843	0.673	0.345	0.374	0.849	0.732
Illness	24	5.066	1.591	2.326	0.976	5.106	1.594	5.049	1.591	4.270	1.484	7.863	2.056	1.977	0.921	4.993	1.508
	36	5.236	1.628	2.152	0.919	5.309	1.629	5.143	1.598	4.293	1.487	8.169	2.122	1.812	0.872	5.147	1.593
	48	5.118	1.600	2.111	0.966	5.381	1.643	5.010	1.574	4.190	1.451	7.144	1.962	1.743	0.856	5.039	1.541
	60	5.349	1.641	2.118	0.968	5.586	1.685	5.164	1.601	4.045	1.430	7.193	1.986	1.816	0.881	5.235	1.601
	Average	5.192	1.615	2.177	0.957	5.346	1.638	5.092	1.591	4.200	1.463	7.592	2.032	1.837	0.883	5.104	1.561
Weather	96	0.581	0.554	0.153	0.202	0.647	0.610	0.202	0.294	0.224	0.312	0.515	0.488	0.146	0.191	0.588	0.561
	192	0.598	0.567	0.194	0.241	0.649	0.607	0.251	0.336	0.273	0.354	0.516	0.488	0.194	0.238	0.604	0.574
	336	0.593	0.558	0.239	0.275	0.674	0.619	0.294	0.364	0.330	0.388	0.505	0.484	0.243	0.275	0.601	0.568
	720	0.611	0.574	0.337	0.344	0.640	0.593	0.364	0.413	0.411	0.433	0.513	0.499	0.318	0.328	0.617	0.582
	Average	0.596	0.563	0.231	0.266	0.653	0.607	0.278	0.352	0.310	0.372	0.512	0.490	0.225	0.258	0.603	0.571

Table 16: MSE and MAE comparison of 8 imaging methods on TSF benchmark datasets. Red numbers indicate the best performance for each dataset.

B.5 Full Results of RQ3: Are the pre-trained parameters in LVMs useful in time series tasks?

Table 17 and Table 18 provide the results of comparing different fine-tuning strategies on TSC and TSF tasks, respectively. In this ablation analysis, we progressively freeze the components of the Transformer blocks in LVMs (Fig. 5) with the following settings: (a) Fine-tune all parameters; (b) Fine-tune all parameters but freeze `CLS` token and `Mask` token; (c) Fine-tune MLP and norm layers only; (d) Fine-tune norm layers only; (e) Freeze all parameters (*i.e.*, zero-shot); and (f) Randomly initialize an LVM and train it from scratch. From Table 17, for TSC, fully fine-tuning all parameters yields the best performance. From Table 18, for TSF, fine-tuning only the norm layer leads to better performance than other settings.

B.6 Full Results of RQ4: How useful are LVMs' architectures?

Table 19 and Table 20 provide the results of comparing LVMs' architecture and two ablation models, w/o-LVM and LVM2Attn, on TSC and TSF tasks, respectively. Fig. 11 illustrates the ablation models. Both models keep the projection layer in LVM encoder. The model w/o-LVM replaces the Transformer blocks with a linear layer. The model LVM2Attn replaces the Transformer blocks with a single multi-head self-attention layer. Other components including input alignment and the

Dataset	(a)	(b)	(c)	(d)	(e)	(f)
UWaveGestureLibrary	88.4	87.5	88.7	81.6	84.0	73.4
SpokenArabicDigits	98.5	98.2	98.4	98.0	98.5	97.0
Handwriting	36.4	35.2	35.5	28.5	27.8	24.3
FaceDetection	67.4	66.3	67.1	65.2	66.7	65.0

Table 17: Accuracy (%) comparison of different fine-tuning strategies for on TSC benchmark datasets. Red numbers indicate the best performance for each dataset.

Fine-tuning Strategy		(a)		(b)		(c)		(d)		(e)		(f)	
Dataset	Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.512	0.448	0.481	0.435	0.477	0.418	0.356	0.383	0.426	0.397	0.412	0.431
	192	0.511	0.453	0.520	0.455	0.526	0.456	0.395	0.406	0.448	0.417	0.462	0.462
	336	0.610	0.512	0.537	0.484	0.584	0.497	0.417	0.424	0.478	0.439	0.489	0.479
	720	0.598	0.523	0.581	0.526	0.539	0.493	0.467	0.463	0.454	0.453	0.536	0.514
	Average	0.558	0.484	0.530	0.475	0.532	0.466	0.409	0.419	0.452	0.427	0.475	0.472
ETTm1	96	0.303	0.334	0.320	0.348	0.306	0.338	0.284	0.333	0.394	0.370	0.323	0.367
	192	0.385	0.385	0.389	0.385	0.385	0.378	0.328	0.363	0.404	0.381	0.344	0.383
	336	0.409	0.403	0.419	0.407	0.420	0.402	0.357	0.384	0.421	0.398	0.375	0.403
	720	0.500	0.461	0.503	0.461	0.474	0.444	0.411	0.417	0.462	0.426	0.446	0.445
	Average	0.399	0.396	0.408	0.400	0.396	0.391	0.345	0.374	0.420	0.394	0.372	0.400
Illness	24	1.888	0.818	1.683	0.789	2.043	0.818	1.977	0.921	2.227	0.971	1.719	0.799
	36	1.542	0.781	1.632	0.801	1.573	0.775	1.812	0.872	2.023	0.932	1.541	0.753
	48	1.682	0.829	1.839	0.845	1.548	0.783	1.743	0.856	1.947	0.920	1.687	0.817
	60	2.012	0.859	1.977	0.921	1.783	0.860	1.816	0.881	1.952	0.939	1.944	0.880
	Average	1.781	0.822	1.783	0.839	1.737	0.809	1.837	0.883	2.037	0.941	1.723	0.812
Weather	96	0.172	0.213	0.174	0.213	0.171	0.208	0.146	0.191	0.274	0.280	0.154	0.201
	192	0.225	0.259	0.233	0.263	0.225	0.256	0.194	0.238	0.284	0.294	0.199	0.245
	336	0.298	0.302	0.296	0.304	0.293	0.303	0.243	0.275	0.311	0.316	0.265	0.292
	720	0.397	0.363	0.397	0.364	0.367	0.361	0.318	0.328	0.364	0.354	0.344	0.350
	Average	0.273	0.284	0.275	0.286	0.264	0.282	0.225	0.258	0.308	0.311	0.241	0.272

Table 18: MSE and MAE comparison of different fine-tuning strategies on TSF benchmark datasets. Red numbers indicate the best performance for each dataset.

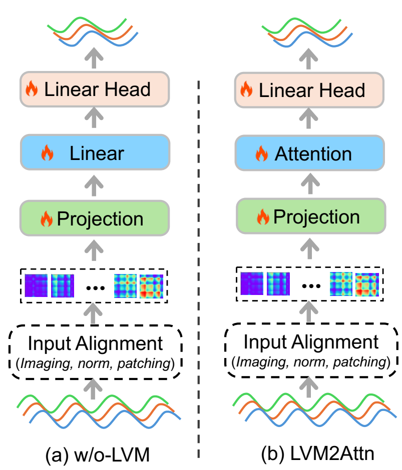

Figure 11: Illustration of LVM's ablation models. (a) is the model W/O-LVM, which replaces the Transformer blocks in LVMs with a linear layer. (b) is the model LVM2ATTN, which replaces the Transformer blocks in LVMs with a single mult-head attention layer.

linear head remain unchanged. In this comparison, all models are trained from scratch without using pre-trained parameters. From Table 19 and Table 20, without pre-trained knowledge, LVMs perform on par with W/O-LVMand LVM2ATTN on both TSC and TSF tasks. However, as demonstrated in Table 17 and Table 18, with pre-training parameters, LVMs outperform both ablation models.

Dataset	LVMs	w/o-LVM	LVM2Attn
UWaveGestureLibrary	73.4	78.6	80.1
SpokenArabicDigits	97.0	96.4	96.5
Handwriting	24.3	22.4	20.7
FaceDetection	65.0	64.1	66.2

Table 19: Accuracy (%) comparison between LVM architecture and ablation models on TSC benchmark datasets. Red numbers indicate the best performance for each dataset.

Model		LVMs		w/o-LVM		LVM2Attn	
Dataset	**Metrics**	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.412	0.431	0.392	0.410	0.391	0.417
	192	0.462	0.462	0.418	0.426	0.414	0.435
	336	0.489	0.479	0.441	0.443	0.438	0.452
	720	0.536	0.514	0.441	0.465	0.469	0.485
	Average	0.475	0.472	0.423	0.436	0.428	0.447
ETTm1	96	0.323	0.367	0.322	0.364	0.298	0.354
	192	0.344	0.383	0.353	0.381	0.338	0.380
	336	0.375	0.403	0.388	0.401	0.376	0.401
	720	0.446	0.445	0.440	0.432	0.416	0.427
	Average	0.372	0.400	0.376	0.395	0.357	0.391
Illness	24	1.719	0.799	2.280	1.034	1.990	0.909
	36	1.541	0.753	2.224	1.018	1.913	0.899
	48	1.687	0.817	2.296	1.039	2.105	0.964
	60	1.944	0.880	2.364	1.052	2.423	1.033
	Average	1.723	0.812	2.291	1.036	2.108	0.951
Weather	96	0.154	0.201	0.188	0.243	0.184	0.240
	192	0.199	0.245	0.226	0.273	0.226	0.271
	336	0.265	0.292	0.270	0.302	0.271	0.303
	720	0.344	0.350	0.336	0.347	0.335	0.346
	Average	0.241	0.272	0.255	0.291	0.254	0.290

Table 20: MSE and MAE comparison between LVM architecture and ablation models on TSF benchmark datasets. Red numbers indicate the best performance for each dataset.

B.7 Full Results of RQ5: Do LVMs capture temporal order of time series?

Four kinds of perturbation, **Sf-All**, **Sf-Half**, **Ex-Half** and **Masking**, are applied to the time series to compare the performance drop of LVMs, w/o-LVM, and LVM2Attn on both TSC and TSF tasks. Table 21 and Table 22 summarize the results. As can be seen, LVMs are more vulnerable to temporal perturbations than the ablation models.

Model		LVMs				w/o-LVM				LVM2Attn			
Dataset	**Perturbation**	Shuffle All	Shuffle Half	Ex-half	Masking	Shuffle All	Shuffle Half	Ex-half	Masking	Shuffle All	Shuffle Half	Ex-half	Masking
UWaveGestureLibrary	Accuracy(%)	17.1	56.2	35.9	62.8	17.1	73.4	0.9	79.4	10.9	73.1	0.9	79.3
	Performance Drop	80.7%	36.4%	59.4%	29.0%	78.2%	6.6%	98.8%	-1.0%	86.4%	8.7%	98.9%	1.0%
SpokenArabicDigits	Accuracy(%)	15.1	68.8	9.9	57.3	48.5	84.4	17.2	93.4	47.7	85.3	17.2	93.0
	Performance Drop	84.7%	30.2%	89.9%	41.8%	49.7%	12.4%	82.2%	3.1%	50.6%	11.6%	82.2%	3.6%
Handwriting	Accuracy(%)	3.1	4.9	1.1	16.0	4.1	5.7	3.7	17.4	2.1	3.4	2.7	16.5
	Performance Drop	91.5%	86.5%	97.0%	56.0%	81.7%	74.6%	83.5%	22.3%	89.9%	83.6%	87.0%	20.3%
FaceDetection	Accuracy(%)	47.7	61.1	61.2	62.4	51.7	57.2	49.5	64.9	51.4	58.7	49.9	64.4
	Performance Drop	29.2%	9.3%	9.2%	7.4%	19.3%	10.8%	22.8%	-1.2%	22.4%	11.3%	24.6%	2.7%

Table 21: Comparison of accuracy (%) and performance drop (%) between LVMs and the ablation models under temporal perturbations on the TSC benchmark datasets. Red numbers indicate the largest performance drop for each dataset.

Model		LVMs								w/o-LVM								LVM2Attn							
Perturbation		Sf-All		Sf-Half		Ex-half		Masking		Sf-All		Sf-Half		Ex-half		Masking		Sf-All		Sf-Half		Ex-half		Masking	
Dataset	Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.747	0.588	0.369	0.393	0.457	0.437	0.551	0.534	0.746	0.582	0.437	0.438	0.483	0.460	0.608	0.559	0.741	0.589	0.442	0.449	0.456	0.448	0.577	0.554
	192	0.734	0.584	0.443	0.446	0.462	0.444	0.578	0.550	0.751	0.590	0.487	0.468	0.481	0.461	0.621	0.567	0.776	0.622	0.515	0.502	0.458	0.452	0.608	0.578
	336	0.733	0.595	0.486	0.469	0.453	0.442	0.612	0.577	0.736	0.591	0.503	0.479	0.470	0.460	0.625	0.574	0.769	0.626	0.537	0.504	0.468	0.467	0.643	0.602
	720	0.765	0.631	0.587	0.549	0.480	0.476	0.664	0.582	0.740	0.613	0.509	0.507	0.472	0.484	0.635	0.597	0.779	0.660	0.554	0.518	0.479	0.491	0.669	0.632
	Avg. Drop	83.8%	43.5%	14.5%	10.4%	14.2%	7.6%	47.5%	34.2%	76.2%	36.4%	14.4%	8.5%	13.0%	7.1%	47.3%	31.8%	79.7%	39.7%	19.5%	10.3%	9.1%	4.0%	46.0%	32.3%
ETTm1	96	0.732	0.561	0.441	0.440	1.127	0.691	0.504	0.508	0.731	0.561	0.441	0.430	0.929	0.629	0.567	0.538	0.779	0.611	0.442	0.447	0.895	0.625	0.577	0.554
	192	0.721	0.562	0.512	0.462	1.146	0.704	0.534	0.519	0.731	0.563	0.463	0.444	0.894	0.618	0.589	0.547	0.768	0.585	0.436	0.442	0.929	0.639	0.525	0.526
	336	0.736	0.568	0.522	0.492	1.163	0.724	0.552	0.533	0.731	0.568	0.485	0.457	0.895	0.622	0.586	0.547	0.730	0.569	0.464	0.454	0.873	0.622	0.552	0.537
	720	0.780	0.587	0.556	0.526	1.221	0.745	0.570	0.547	0.753	0.582	0.529	0.484	0.919	0.636	0.616	0.562	0.772	0.721	0.743	0.585	0.939	0.656	0.771	0.628
	Avg. Drop	118.4%	53.0%	48.2%	28.4%	242.3%	92.2%	58.4%	41.4%	98.4%	44.6%	28.3%	15.2%	145.3%	59.3%	58.5%	39.5%	117.1%	59.3%	44.8%	23.2%	158.3%	63.4%	70.3%	44.0%
Illness	24	4.794	1.578	2.426	1.064	2.465	1.045	4.169	1.386	5.220	1.674	3.091	1.251	2.529	1.098	4.394	1.507	4.712	1.613	3.449	1.287	2.942	1.219	4.768	1.572
	36	4.719	1.572	2.240	1.006	2.256	0.995	4.128	1.372	4.966	1.634	3.181	1.281	2.505	1.095	4.388	1.486	4.240	1.523	3.517	1.132	2.648	1.136	4.683	1.533
	48	4.665	1.561	2.108	0.964	2.157	0.974	4.113	1.373	4.685	1.583	3.240	1.294	2.487	1.089	4.428	1.480	4.179	1.515	3.615	1.359	2.463	1.070	4.689	1.540
	60	5.094	1.622	2.138	0.962	2.161	0.999	4.374	1.422	4.947	1.632	3.464	1.335	2.648	1.129	4.574	1.521	4.349	1.523	3.597	1.352	2.629	1.099	4.940	1.578
	Avg. Drop	162.8%	79.5%	21.3%	13.2%	23.0%	13.7%	128.9%	57.4%	116.4%	57.5%	41.6%	24.6%	11.0%	6.5%	94.1%	44.7%	109.1%	62.9%	69.3%	34.8%	27.9%	19.5%	127.8%	64.0%
Weather	96	0.258	0.316	0.162	0.211	0.329	0.351	0.278	0.371	0.261	0.312	0.189	0.246	0.295	0.329	0.290	0.379	0.261	0.313	0.189	0.246	0.298	0.331	0.290	0.380
	192	0.283	0.329	0.206	0.249	0.336	0.354	0.296	0.371	0.291	0.331	0.230	0.276	0.320	0.342	0.315	0.392	0.288	0.329	0.232	0.280	0.315	0.339	0.315	0.393
	336	0.318	0.349	0.260	0.291	0.358	0.365	0.327	0.398	0.320	0.349	0.286	0.319	0.334	0.351	0.346	0.411	0.319	0.347	0.278	0.312	0.339	0.353	0.339	0.405
	720	0.396	0.411	0.363	0.357	0.391	0.388	0.384	0.439	0.370	0.380	0.341	0.354	0.382	0.382	0.376	0.428	0.371	0.381	0.341	0.353	0.387	0.385	0.376	0.428
	Avg. Drop	44.50%	38.97%	9.57%	7.44%	67.20%	45.88%	49.58%	57.17%	24.06%	18.68%	2.43%	2.49%	33.98%	21.74%	33.42%	39.75%	24.43%	19.11%	2.44%	2.70%	35.49%	22.70%	33.58%	40.18%

Table 22: Comparison of performance drop (%) between the LVM and its ablated variants under time series perturbations on the TSF benchmark datasets. Red numbers indicate the largest performance drop among the three variants.

B.8 Full Results of RQ7: Which component of LVMs contributes more to forecasting

Table 23 provides the detailed results on MSE and MAE of the two ablations, **Enc w/o Dec** and **Dec w/o Enc**, of self-supervised LVMs on TSF benchmark datasets. From Table 23, **Enc w/o Dec** shows inferior performance to **Dec w/o Enc**, highlighting the importance of the pre-trained decoders of LVMs in TSF.

Model		MAE						SimMIM					
		Pre-trained		**Enc w/o Dec**		**Dec w/o Enc**		**Pre-trained**		**Enc w/o Dec**		**Dec w/o Enc**	
Dataset	**Metrics**	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.356	0.383	0.420	0.423	0.396	0.401	0.362	0.383	0.466	0.426	0.412	0.418
	192	0.395	0.406	0.445	0.446	0.399	0.414	0.407	0.412	0.496	0.455	0.457	0.446
	336	0.417	0.424	0.489	0.484	0.441	0.433	0.422	0.417	0.499	0.474	0.581	0.520
	720	0.467	0.463	0.582	0.543	0.426	0.451	0.462	0.455	0.505	0.481	0.564	0.526
	Average	0.409	0.419	0.484	0.474	0.416	0.425	0.413	0.417	0.492	0.459	0.504	0.478
ETTm1	96	0.284	0.333	0.324	0.363	0.295	0.335	0.311	0.350	0.320	0.347	0.299	0.348
	192	0.328	0.363	0.361	0.387	0.330	0.364	0.335	0.367	0.377	0.377	0.344	0.378
	336	0.357	0.384	0.398	0.414	0.365	0.388	0.356	0.382	0.411	0.401	0.403	0.419
	720	0.411	0.417	0.446	0.440	0.409	0.416	0.400	0.413	0.468	0.442	0.431	0.433
	Average	0.345	0.374	0.382	0.401	0.350	0.376	0.351	0.378	0.394	0.392	0.369	0.395
Illness	24	1.977	0.921	1.946	0.842	1.774	0.841	1.934	0.902	2.314	0.944	2.034	0.899
	36	1.812	0.872	1.981	0.895	1.918	0.876	1.754	0.825	2.434	1.045	2.198	0.983
	48	1.743	0.856	1.967	0.855	2.061	0.943	1.715	0.867	2.008	0.869	2.209	0.960
	60	1.816	0.881	1.956	0.858	1.969	0.950	1.673	0.877	1.979	0.865	2.275	0.997
	Average	1.837	0.883	1.963	0.863	1.931	0.903	1.769	0.868	2.184	0.931	2.179	0.960
Weather	96	0.146	0.191	0.168	0.210	0.155	0.201	0.148	0.196	0.166	0.208	0.150	0.200
	192	0.194	0.238	0.237	0.263	0.209	0.248	0.196	0.243	0.228	0.257	0.199	0.246
	336	0.243	0.275	0.299	0.306	0.274	0.298	0.244	0.276	0.294	0.297	0.251	0.284
	720	0.318	0.328	0.396	0.372	0.378	0.361	0.340	0.340	0.382	0.357	0.343	0.342
	Average	0.225	0.258	0.275	0.288	0.254	0.277	0.232	0.264	0.268	0.280	0.236	0.268

Table 23: MSE and MAE comparison of self-supervised LVMs with either the pre-trained encoder (**Dec w/o Enc**) or decoder (**Enc w/o Dec**) excluded on TSF benchmark datasets.

B.9 Full Results of RQ8: Will period-based imaging method induce any bias?

Fig. 12 provides the forecasting performance of an LVM (*i.e.*, MAE) in terms of metrics MAE *w.r.t.* segment length that varies from $\frac{1}{6}L$ to $\frac{12}{6}L$. The LVM generally achieves the best performance when segment length is a multiple of the period, *i.e.* L or $2L$, which is caused by the inductive bias as discussed in RQ8 In §4.4.

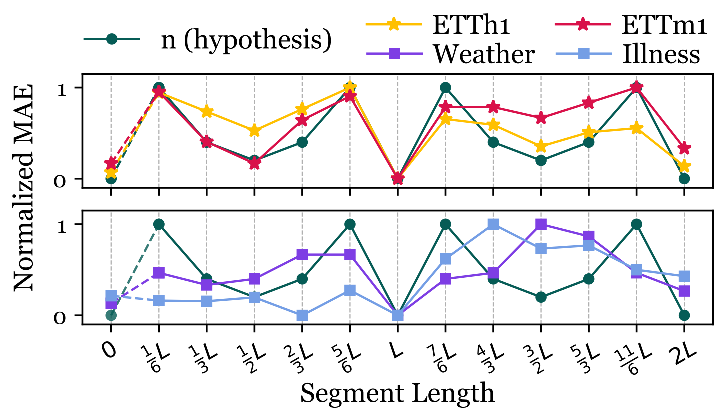

Figure 12: Forecasting performance (MAE) of an LVM *w.r.t.* varying segment length used in UVH imaging. n (green) estimates the difficulty of forecasting.

B.10 Full Results of RQ6: What are the computational costs of LVMs?

Fig. 13 presents the accuracy and inference efficiency comparison between LVMs and the two best-performing baselines on TSC task. Fig. 14 (Fig. 15) presents the MSE (MAE) and inference efficiency comparisons between LVMs and the two best-performing baselines on TSF task. In general, LVMs can yield improved performance with higher costs of inference time.

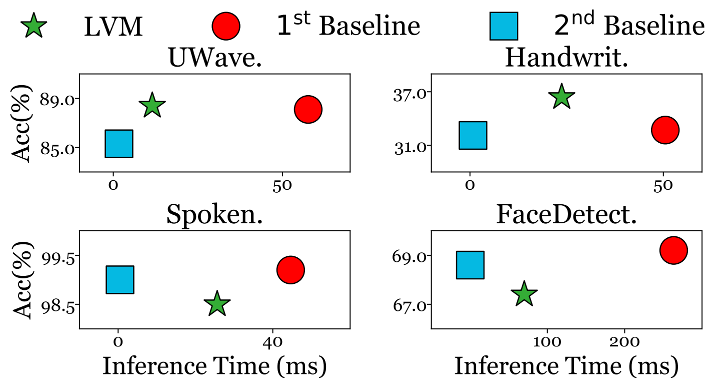

Figure 13: Accuracy *vs.* inference time of the compared methods on TSC benchmark datasets. Green marker stands for LVM, Red marker stands for GPT4TS and Blue marker stands for TimesNet.

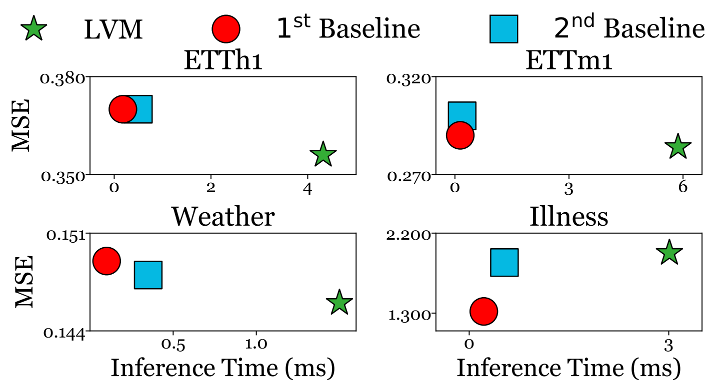

Figure 14: MSE *vs.* inference time of the compared methods on TSF benchmark datasets. Green marker stands for LVM, Red marker stands for PatchTST and Blue marker stands for GPT4TS.

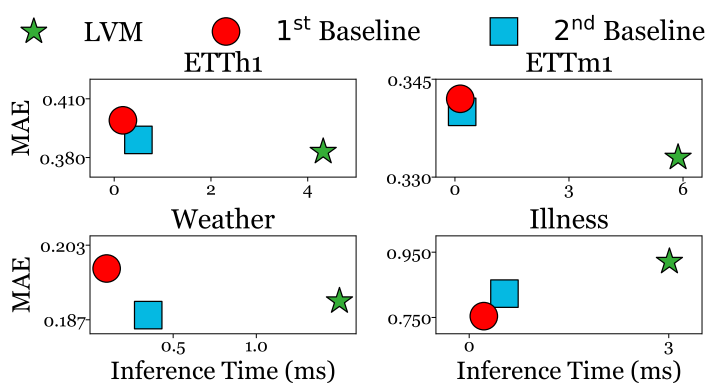

Figure 15: MAE *vs.* inference time of the compared methods on TSF benchmark datasets. Green marker stands for LVM, Red marker stands for PatchTST and Blue marker stands for GPT4TS.

B.11 Full Results of RQ9: Can LVMs make effective use of look-back windows?

Table 24 presents the MSE and MAE performance of LVMs across varying look-back window lengths, ranging from 48 to 2304. As discussed in RQ9, LVMs exhibits limited ability in fully leveraging the information of look-back window when the window length exceeds approximately 1000 time steps. The Illness dataset is omitted in Table 24 because its time series are of short lengths, with only 966 time steps in total.

Look-back Window		48		96		192		336		720		1152		1728		2304	
Dataset	Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.376	0.395	0.373	0.390	0.364	0.383	0.356	0.383	0.347	0.375	0.347	0.376	0.344	0.376	0.373	0.402
	192	0.440	0.431	0.424	0.418	0.411	0.412	0.395	0.406	0.385	0.405	0.384	0.402	0.391	0.408	0.399	0.417
	336	0.474	0.450	0.471	0.445	0.456	0.437	0.417	0.424	0.408	0.418	0.410	0.418	0.395	0.413	0.408	0.423
	720	0.485	0.477	0.482	0.471	0.469	0.465	0.467	0.463	0.468	0.460	0.432	0.440	0.425	0.442	0.424	0.442
	Average	0.444	0.438	0.438	0.431	0.425	0.424	0.409	0.419	0.402	0.415	0.393	0.409	0.389	0.410	0.401	0.421
ETTm1	96	0.443	0.413	0.316	0.353	0.304	0.345	0.284	0.333	0.279	0.324	0.280	0.332	0.277	0.322	0.285	0.326
	192	0.476	0.431	0.373	0.390	0.333	0.365	0.328	0.363	0.322	0.358	0.321	0.361	0.321	0.355	0.318	0.350
	336	0.512	0.457	0.385	0.400	0.370	0.390	0.357	0.384	0.356	0.381	0.362	0.383	0.352	0.378	0.346	0.374
	720	0.574	0.489	0.449	0.438	0.426	0.429	0.411	0.417	0.411	0.414	0.399	0.413	0.411	0.414	0.407	0.416
	Average	0.501	0.448	0.381	0.395	0.358	0.382	0.345	0.374	0.342	0.369	0.341	0.372	0.340	0.367	0.339	0.367
Weather	96	0.200	0.237	0.167	0.209	0.152	0.196	0.146	0.191	0.142	0.188	0.144	0.194	0.143	0.193	0.141	0.195
	192	0.236	0.267	0.212	0.249	0.200	0.240	0.194	0.238	0.188	0.235	0.189	0.237	0.195	0.242	0.200	0.253
	336	0.293	0.307	0.268	0.290	0.254	0.280	0.243	0.275	0.247	0.281	0.242	0.279	0.272	0.302	0.278	0.307
	720	0.370	0.358	0.346	0.340	0.330	0.333	0.318	0.328	0.334	0.341	0.332	0.339	0.344	0.349	0.372	0.357
	Average	0.275	0.292	0.248	0.272	0.234	0.262	0.225	0.258	0.228	0.261	0.227	0.262	0.239	0.272	0.248	0.278

Table 24: The MSE and MAE performance of LVMs across different look-back window lengths on TSF benchmark datasets.

C Proof of Lemma 1

In this section, we provide the proof for Lemma 1.

Proof. Given $\mathbf{x}$ is perfectly periodic, $x_t = x_{t+\alpha\cdot L}$ holds when $\alpha \in \mathbb{N}^+$ and L is the period. The smallest number of segments n before any segment reoccurs, *i.e.*, $\mathbf{x}_t = \mathbf{x}_{t+n\cdot(i/k)L}$, indicates $n \cdot (i/k) \in \mathbb{N}^+$. Hence, the proof of Lemma 1 is equivalent to prove $n = \frac{k}{\text{GCD}(i,k)}$ as the smallest natural number such that k divides $n \cdot i$, denoted as $k \mid n \cdot i$.

Set $d = \text{GCD}(i, k)$ as the greatest common divisor of i and k. The following is based on the definition of greated common divisor:

$$i = d \cdot i' \tag{3}$$

$$k = d \cdot k' \tag{4}$$

$$\text{GCD}(i', k') = 1 \tag{5}$$

where $i', k' \in \mathbb{N}^+$. As k divides $n \cdot i$, we have

$$\begin{aligned} k \mid n \cdot i &\Rightarrow d \cdot k' \mid d \cdot n \cdot i' \\ &\Rightarrow k' \mid n \cdot i' \\ &\Rightarrow k' \mid n \end{aligned} \tag{6}$$

The first step in Eq. (6) is expanded with Eq. (3) and Eq. (4). The second step cancels the common factor d from both sides of with the divisibility relation unchanged. The last step follows Eq. (5). To satisfy Eq. (6), the smallest n is $n = k'$. Finally, expand k' with Eq. (4), we reach

$$n = k' = \frac{k}{d} = \frac{k}{\text{GCD}(i, k)}$$

□

D Visualization Results

D.1 Visualization of GAF on TSC Task

To have a sense about what temporal patterns can be recognized by LVMs for TSC, we visualize the images of GAF method on the Handwriting and UWaveGestureLibrary datasets in Fig. 16 and Fig.

17, respectively. The examples are randomly sampled from five different classes on both datasets. From Fig. 16 and Fig. 17, we can observe clear visual patterns that distinguish the GAF images from different classes, which highlight the effectiveness of GAF as a way to encode time series for LVMs to process for TSC.

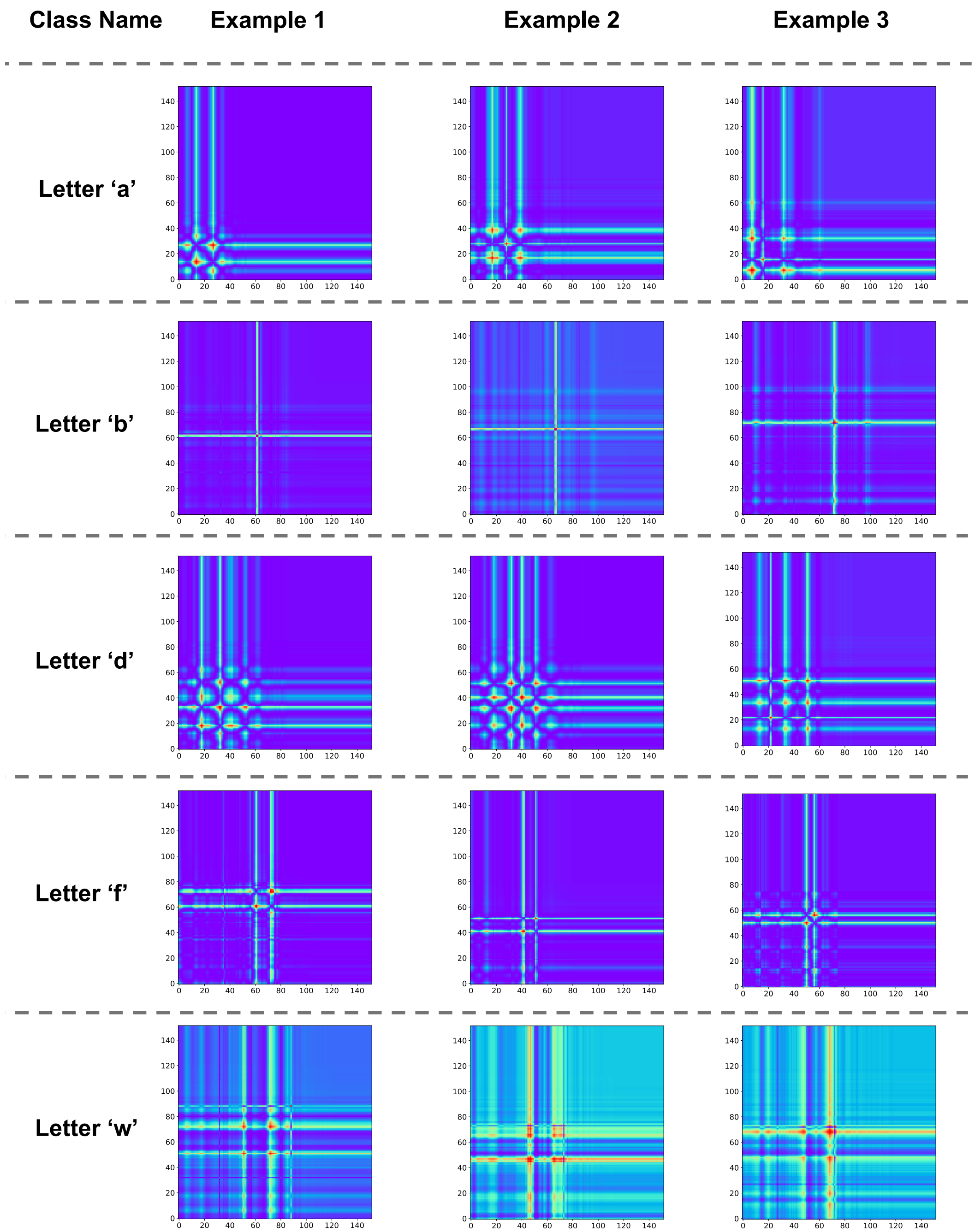

Figure 16: Examples of GAF images on the first channel of multivariate time series with 152 time steps randomly drawn from five classes in the Handwriting dataset.

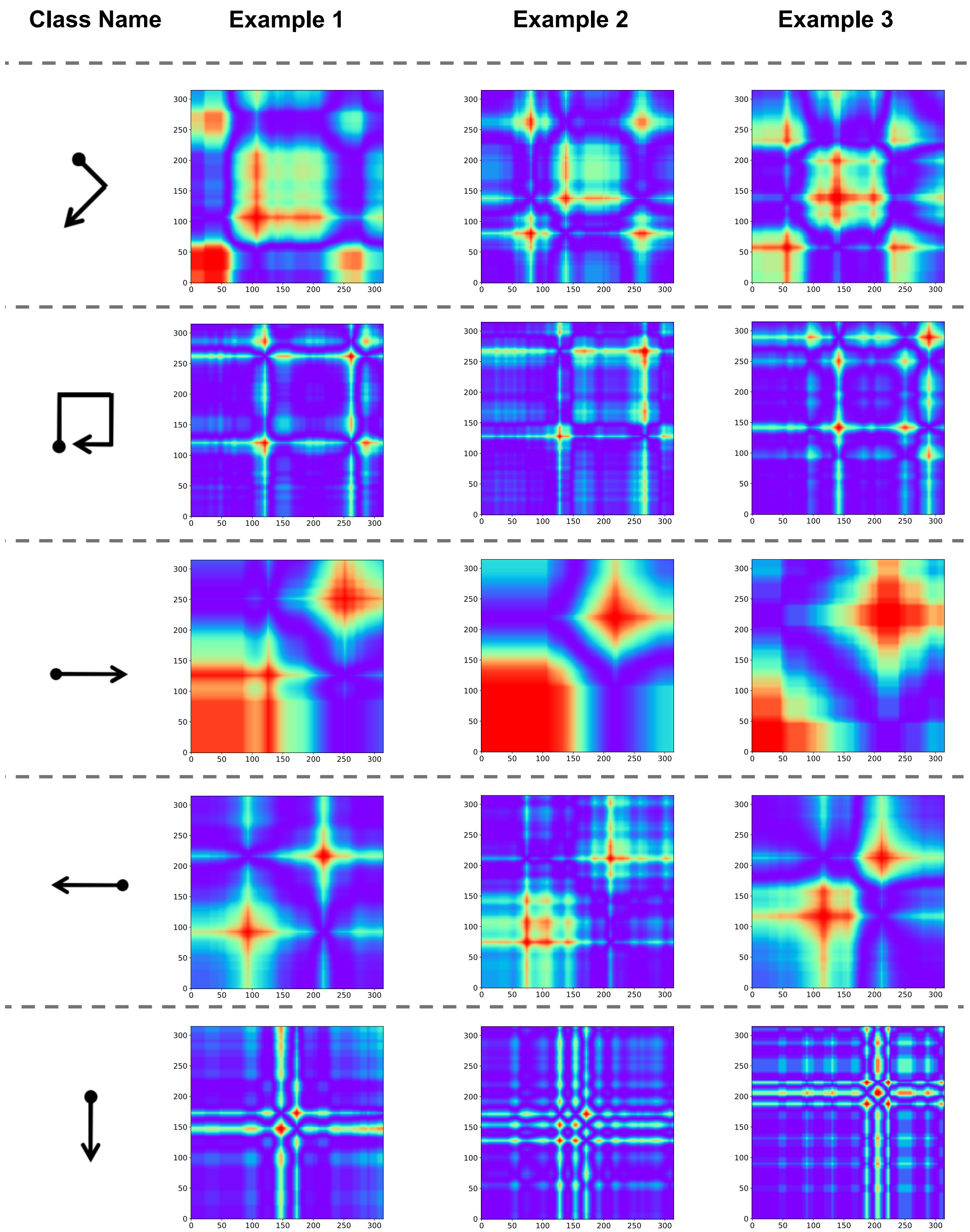

Figure 17: Examples of GAF images on the first channel of multivariate time series with 336 time steps randomly drawn from five classes in the UWaveGestureLibrary dataset.

D.2 Illustration of An Inductive Bias of LVMs during TSF

As discussed in RQ8, the imaging method UVH can induce an inductive bias to LVMs in TSF toward "forecasting periods" by rendering them to combine the past segments to infer future. To illustrate this, Fig. 18 and Fig. 19 visualize two random examples with varying segment lengths from one period (24 time steps) to two periods (48 time steps) from ETTh1 and Traffic datasets. The blue lines represent the time series in look-back window, the red lines represent the ground truth in prediction horizon, and the green lines represent the forecasted time series by LVMs. The results demonstrate that LVMs perform best when the segment length aligns with the period of the time series, while the performance degrades when the segment length shifts from the period. This implies the inductive bias of combining the past periods as forecasts by LVMs with UVH for TSF.

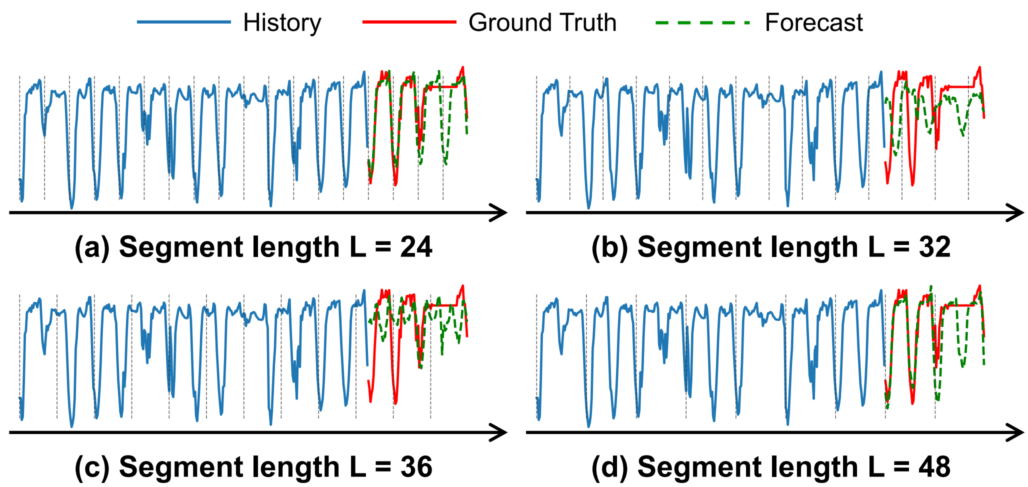

Figure 18: Visualization of LVM's inductive bias during TSF on a random example from the ETTh1 dataset (period is 24 time steps). From (a) to (d), the segment length vary within {24, 32, 36, 48}.

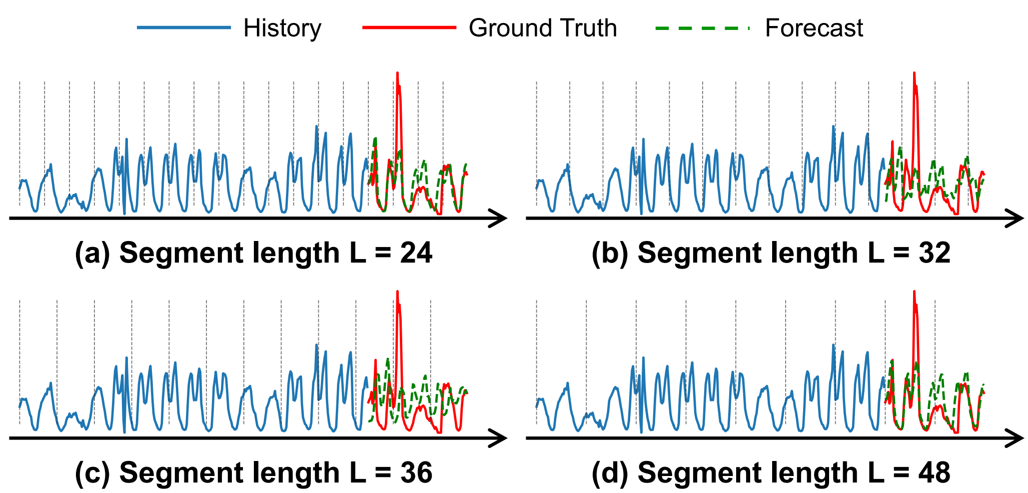

Figure 19: Visualization of LVM's inductive bias during TSF on a random example from the Traffic dataset (period is 24 time steps). From (a) to (d), the segment length vary within {24, 32, 36, 48}.