Empowering Large Language Models for Time Series Forecasting with Patterns and Semantics

Jialiang Tang[1,2], Shuo Chen[3,5], Chen Gong[1*], Jing Zhang[4], Dacheng Tao[2*]
[1]School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai, China
[2] College of Computing and Data Science, Nanyang Technological University, Singapore
[3]School of Intelligence Science and Technology, Nanjing University, Nanjing, China
[4]School of Computer Science, Wuhan University, Wuhan, China
[5]Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan
chen.gong@sjtu.edu.cn, dacheng.tao@ntu.edu.sg

***Abstract*—Time Series Forecasting (TSF) is critical in many real-world domains like financial planning and health monitoring. Recent studies have revealed that Large Language Models (LLMs), with their powerful in-contextual modeling capabilities, hold significant potential for TSF. However, existing LLM-based methods usually perform suboptimally because they neglect the inherent characteristics of time series data. Unlike the textual data used in LLM pre-training, the time series data is semantically sparse and comprises distinctive temporal patterns. To address this problem, we propose LLM-PS to empower the LLM for TSF by learning the fundamental *Patterns* and meaningful *Semantics* from time series data. Our LLM-PS incorporates a new multi-scale convolutional neural network adept at capturing both short-term fluctuations and long-term trends within the time series. Meanwhile, we introduce a time-to-text module for extracting valuable semantics across continuous time intervals rather than isolated time points. By integrating these patterns and semantics, LLM-PS effectively models temporal dependencies, enabling a deep comprehension of time series and delivering accurate forecasts. Intensive experimental results demonstrate that LLM-PS achieves state-of-the-art performance in both short- and long-term forecasting tasks, as well as in few- and zero-shot settings. Code is available at https://github.com/tangjialiang97/LLMPS.**

***Index Terms*—Time Series Forecasting, Large Language Models, Temporal Patterns**

I. Introduction

Time Series Forecasting (TSF) plays a crucial role in various real-world applications, such as weather forecasting [1] and energy consumption prediction [34]. To achieve reliable TSF, traditional deep learning methods [41, 49, 51] usually rely on domain-specific expertise to design customized models tailored to individual tasks. However, time series data often vary significantly across different domains [48]. For example, stock prices in the financial market frequently fluctuate in short intervals, while temperature readings from the weather station typically evolve gradually over days. Therefore, those task-specific models struggle to generalize effectively across different applications [18]. Moreover, these models are commonly trained from scratch and prone to overfitting in practical scenarios due to limited training data availability [38].

* Corresponding author

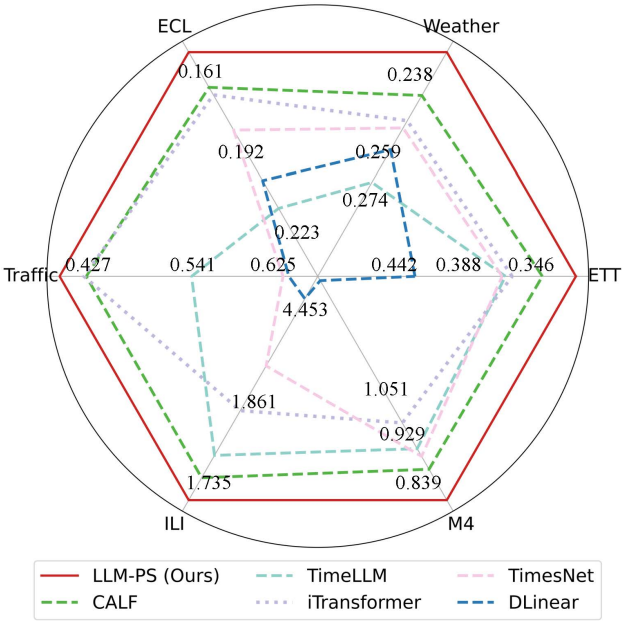


Fig. 1: Performance (average MSE over 4 prediction lengths) of our proposed LLM-PS, LLM-based methods [18, 21], and conventional deep learning methods [22, 41, 44].

Recently, Large Language Models (LLMs), such as GPT [3] and Llama [33], have achieved great success in natural language processing. These LLMs are pre-trained on large-scale serialized textual datasets, endowing them with strong capabilities in both contextual modeling and generalization. In general, time series and textual data share two primary similarities: 1) *Sequentiality*, both time series and textual data consist of ordered sets of elements; 2) *Contextual dependency*, the meaning of a textual sentence relies on its context, and the value at the current time point is driven by its historical data. Building on these parallels, pioneering works [4, 18, 21] attempt to fine-tune powerful LLMs for time series forecasting. Among them, Liu et al. [21] align the distributions of time series and textual data to enhance the LLM's effectiveness in TSF. TimeLLM [18] bridges the modalities of time series

and textual data by reprogramming time series data with text prototypes, thereby unlocking the TSF performance of LLMs.

Despite significant advancements in existing LLM-based methods for TSF, they generally prioritize aligning textual and time series data while ignoring the inherent characteristics of time series data, resulting in suboptimal performance. First, time series data exhibit *diverse temporal patterns* rarely present in textual data, including regular short-term periodic fluctuations and persistent long-term trends that evolve over time [42, 51]. Second, the *semantic information* is spare in time series data [6], usually requiring a prolonged period to convey specific semantics like "rapid increase" or "sudden drop". In comparison, words in textual data generally express explicit meaning, such as "fast" or "slow". Therefore, identifying fundamental temporal patterns and specific semantic information within time series data is crucial to guide LLMs for reliable time series prediction.

In this paper, we propose a novel LLM fine-tuning framework called LLM-PS, which enhances time series forecasting by leveraging the *Patterns* and *Semantics* in time series data. LLM-PS comprises two pivotal modules: the *Multi-Scale Convolutional Neural Network* (MSCNN), designed to capture the intrinsic temporal patterns, and the *Time-to-Text* semantic information extractor (T2T). Specifically, MSCNN extracts multi-scale features with varying receptive fields by hierarchical stacked convolutional layers. Features with small receptive fields primarily capture short-term patterns (*i.e.*, periodicity fluctuations), while those with large receptive fields focus on long-term patterns (*i.e.*, global trends). To further cope with short-term and long-term patterns, we decouple them from multi-scale features based on the wavelet transform and enhance them via global-to-local and local-to-global assembling. Meanwhile, T2T extracts valuable semantics by precisely predicting labels for time-series patches under a high masking ratio. Subsequently, the temporal patterns and semantics are integrated through feature transfer and input into the LLM to generate time series. Thanks to its ability to effectively handle diverse temporal patterns and accurately extract semantically rich information, LLM-PS comprehensively understands time series data, consistently achieving state-of-the-art (SOTA) performance, as shown in Fig. 1.

- We recognize that time series data exhibits intrinsic characteristics that differ from those in textual data used in LLM pre-training and propose a novel TSF framework to leverage these distinctive properties to derive the LLM for reliable time series forecasting.
- We design new MSCNN and T2T modules tailored for effectively handling the diverse temporal patterns and accurately extracting semantic information, and thus enhancing the LLM's understanding of the input time series during forecasting.
- Our LLM-PS consistently achieves SOTA performance across a variety of mainstream time series prediction tasks, especially in few-shot and zero-shot scenarios. Furthermore, our model is highly efficient while robust to noise compared with other popular methods.

II. Related Works

A. Time Series Forecasting

Time series forecasting aims to predict future values of a series based on historical data, serving as a crucial capability in industries like finance management [27], weather forecasting [1], and energy consumption prediction [50]. Recently, TSF methods have evolved from traditional statistical models [26, 32] to sophisticated deep learning models [37, 41]. Deep learning methods generally utilize Recurrent Neural Networks (RNNs) [30], Convolutional Neural Networks (CNNs) [41], Transformers [42], and Multi-Layer Perceptrons (MLPs) [37] as their backbones. By leveraging domain expertise, they can perform well on specific tasks. Nonetheless, their real-world applicability is usually constrained by the variability of temporal patterns across domains [18].

To address these challenges, some researchers [21] have turned to LLMs for TSF and achieved great success. Current methods primarily focus on bridging the gap between time series and textual modalities. For example, TimeLLM [18] further enhances guidance for LLM by incorporating domain information, instructions, and data statistics within the prompts. Instead of designing prompts, [7] directly train a codebook to convert continuous time series into discrete input embeddings by mapping them to the most similar codewords in the codebook. Similarly, [29] apply K-Means clustering to time series embeddings and construct a codebook [35] by the cluster centroids. Additionally, CALF [21] and TimeCMA [20] train separate LLM branches for time series and textual data, and then align their features together for subsequently forecasting.

Despite advancements in LLM-based TSF methods that focus on aligning textual and time series data, recent research [31] suggests that these approaches often inadequately capture the inherent sequential dependencies within time series, resulting in suboptimal forecasting accuracy. Our LLM-PS effectively handles these properties of temporal patterns and semantics, thereby achieving reliable performance.

B. Temporal Patterns Learning

Time series data comprises short-term fluctuations and long-term trends [46]. To capture these temporal patterns, existing methods [19, 37] convert the original inputs to multiple time series inputs with varying scales through pooling operations with various window sizes. Consequently, the model can learn short-term periods and long-term trends from low-scale and high-scale inputs, respectively. However, these methods incur prohibitive computational overheads due to the intricate processing of numerous scaled signals. To address this, we introduce a new MSCNN that enables efficiently generating multi-scale features with diverse temporal patterns using a single forward.

Besides to construct multi-scale signals, some methods [42, 43] directly separate temporal patterns from time series. Earlier studies [42, 43] employ average pooling to extract long-term

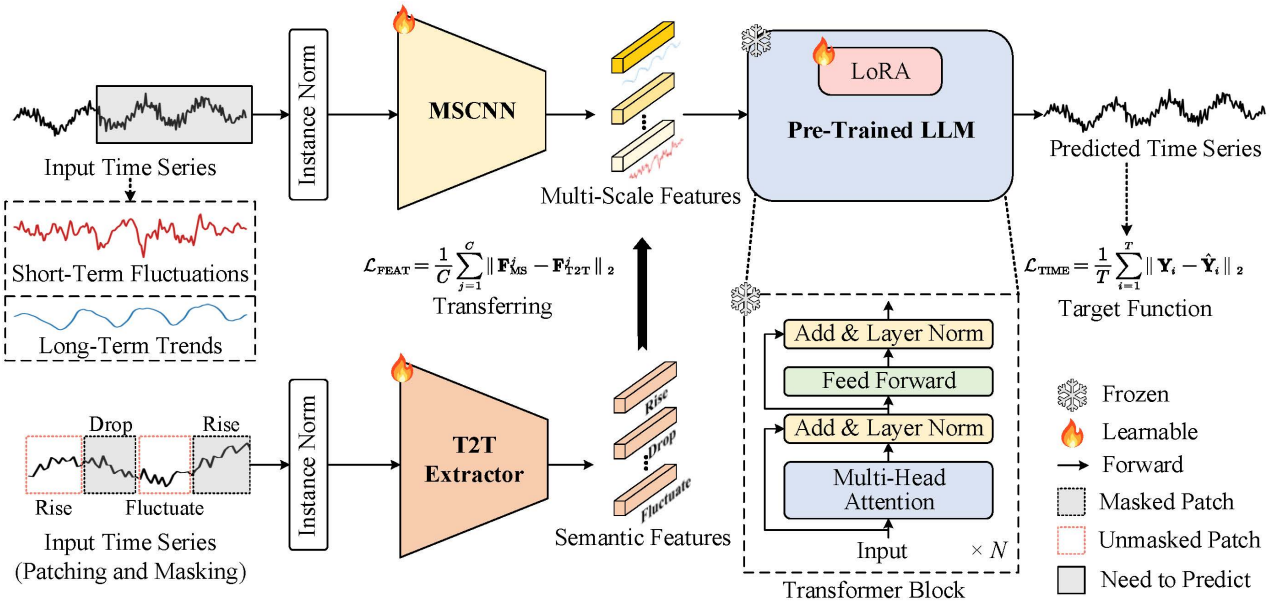


Fig. 2: An overview of our proposed LLM-PS. Our LLM-PS incorporates a Multi-Scale Convolutional Neural Network (MSCNN) and Time-to-Text (T2T) semantics extractor. Specifically, for input time series $\mathbf{X}$, MSCNN constructs multi-scale features $\mathbf{F}_{\text{MS}}$ with various receptive fields (darker colors indicate larger receptive fields), thereby capturing localized short-term fluctuations and broader long-term trends. T2T extracts features $\mathbf{F}_{\text{T2T}}$ with meaningful semantics to promote the LLM to precisely understand the input time series. Finally, the diverse temporal patterns and rich semantics are integrated via feature transferring and input into the LLM to generate precise time series $\hat{\mathbf{Y}}$.

trends from time series data while considering the remaining segments as short-term patterns. Recent studies [37, 51] highlighted that low-frequency and high-frequency components in the frequency domain correspond to long-term and short-term patterns, respectively. Therefore, they utilize the Fourier transform [9] to isolate them. In this paper, we decouple short-term and long-term patterns based on the wavelet transform [45]. Unlike the aforementioned methods that rely on average pooling in the temporal domain or Fourier transform in the frequency domain, our method concurrently learns from both the temporal and frequency domains. Therefore, our approach can accurately decompose short-term and long-term components.

III. Approach

Time series forecasting aims to predict the series $\mathbf{Y} \in \mathbb{R}^{T\times V}$ for the next T time steps given the H time steps historical observations $\mathbf{X} \in \mathbb{R}^{H\times V}$, where V denotes the number of variables[1]. As shown in Fig. 2, our LLM-PS incorporates a new MSCNN (detailed in Sections III-A&III-B) to learn multi-scale features from the input time series, which captures both short-term and long-term patterns. Meanwhile, within LLM-PS, the T2T module (described in Section III-C) enriches the multi-scale features by the semantic information extracted from the input time series. Consequently, the multi-scale features with diverse temporal dependencies and valuable semantics are input into the LLM to facilitate accurate future time series $\hat{\mathbf{Y}} \in \mathbb{R}^{T\times V}$.

[1]This is also referred to as a "channel" in some prior works.

A. Multi-Scale Convolutional Neural Network

Real-world time series data is inherently complex, manifesting short-term and long-term patterns [8, 46], both of which are critical for accurate forecasting. Short-term patterns reflect localized fluctuations and periodic dynamics, while long-term patterns encapsulate broader trends that signal future trajectories. In vanilla CNNs, each convolutional layer has a fixed receptive field, which limits its output features to a narrow temporal scope with only a single temporal pattern. To capture diverse temporal patterns, we follow classic CNNs [13, 15] and design a new MSCNN stacked with bottleneck blocks. As depicted in Fig. 3, each MSCNN block learns multi-scale features by parallel branches. Features with smaller receptive fields focus on periodic fluctuations, whereas those with larger receptive fields concentrate on overarching trends.

In each MSCNN block, the C channels input features $\mathbf{F}_{\text{in}} \in \mathbb{R}^{C\times V}$ first undergo 1×1 convolutional layer before partitioned to B branches $\{\mathbf{F}_1, \ldots, \mathbf{F}_B\}$, where $\mathbf{F}_i \in \mathbb{R}^{C/B\times V}$ (batch dimension is omitted for simplify description). Then, B branches features are recursively fed into their respective 3×3 convolutional layer and added with the output of the preceding branches (except for $\mathbf{F}_1$), as follows:

$$\bar{\mathbf{F}}_i = \begin{cases} \mathbf{Conv}_i\left(\mathbf{F}_i\right), & i = 1, \\ \mathbf{Conv}_i\left(\mathbf{F}_i + \bar{\mathbf{F}}_{i-1}\right), & 1 < i \le B. \end{cases} \tag{1}$$

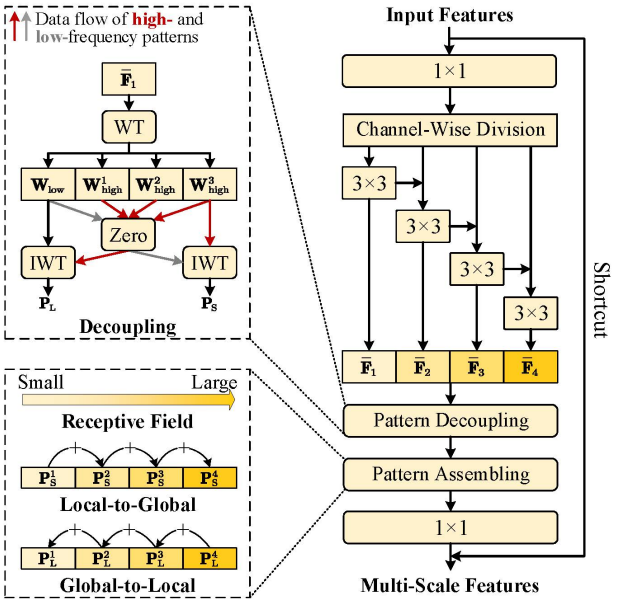


Fig. 3: The diagram of our MSCNN block. The divided features are initially fed into their related 3×3 convolutional layers to obtain features (*e.g.*, $\bar{\mathbf{F}}_1$) with various receptive fields. Then, these features are decoupled into long-term (*e.g.*, $\mathbf{P}_{\mathrm{L}}^1$) and short-term patterns (*e.g.*, $\mathbf{P}_{\mathrm{S}}^1$) using the Wavelet Transform (WT) and Inverse Wavelet Transform (IWT). Subsequently, the long-term and short-term patterns are enhanced through global-to-local and local-to-global assembling, respectively. Finally, the improved patterns are combined and passed through a 1×1 convolution to obtain the multi-scale features.

The receptive fields of features in $\{\bar{\mathbf{F}}_1, \ldots, \bar{\mathbf{F}}_B\}$ increase sequentially as each $\bar{\mathbf{F}}_i$ (if $i > 1$) aggregates information from preceding branches. Finally, these features are concatenated together and fused through a 1×1 convolution layer to derive the output features $\mathbf{F}_{\text{out}}$:

$$\mathbf{F}_{\text{out}} = \text{Conv}_{1\times1}(\text{Concate}(\{\bar{\mathbf{F}}_1, \ldots, \bar{\mathbf{F}}_B\}) + \mathbf{F}_{\text{in}}, \quad (2)$$

where $\mathbf{F}_{\text{in}}$ is added in the output features via shortcut. Afterward, $\mathbf{F}_{\text{out}}$ is input into the subsequent MSCNN blocks to produce the multi-scale features $\mathbf{F}_{\text{MS}}$ for the LLM.

B. Temporal Patterns Decoupling and Assembling

In the previous section, we show how MSCNN captures diverse temporal patterns by learning multi-scale features. For a time series, its high-frequency and low-frequency components can effectively represent the corresponding short-term and long-term temporal patterns, respectively [19, 37]. Therefore, we introduce a novel patterns decoupling-assembling mechanism based on the wavelet transform [45] to further refine temporal patterns in the multi-scale features.

Initially, the features $\{\bar{\mathbf{F}}_1, \ldots, \bar{\mathbf{F}}_B\}$ output by B branches in the MSCNN block are decoupled into low-frequency component $\mathbf{W}_{\text{low}}$ and high-frequency component $\mathbf{W}_{\text{high}}$ using the wavelet transform [45] WT(·):

$$\mathbf{W}_{\text{low}}^b, \{\mathbf{W}_{\text{high\_}i}^b\}_{i=1}^w = \text{WT}(\bar{\mathbf{F}}_b, w), \quad b \in \{1, \ldots, B\}, \quad (3)$$

where w denotes the decomposition levels. Subsequently, the short-term pattern $\mathbf{P}_{\mathrm{S}}$ and long-term pattern $\mathbf{P}_{\mathrm{L}}$ are constructed by the inverse wavelet transform IWT(·):

$$\begin{cases} \mathbf{P}_{\mathrm{S}}^b = \text{IWT}(\text{Zero}(\mathbf{W}_{\text{low}}^b), \{\mathbf{W}_{\text{high\_}i}^b\}_{i=1}^w), \\ \mathbf{P}_{\mathrm{L}}^b = \text{IWT}(\mathbf{W}_{\text{low}}^b, \{\text{Zero}(\mathbf{W}_{\text{high\_}i}^b)\}_{i=1}^w), \end{cases} \quad (4)$$

where Zero(·) operation generates features matching the input dimensions but filled with zeros.

For features in $\{\bar{\mathbf{F}}_1, \ldots, \bar{\mathbf{F}}_B\}$, their receptive fields expand sequentially. Features with smaller receptive fields more effectively capture local periodic fluctuations, whereas those with larger receptive fields focus on broader global trends. Therefore, short-/long-term patterns $\mathbf{P}_{\mathrm{S}}$/$\mathbf{P}_{\mathrm{L}}$ are reinforced in local-to-global/global-to-local assembling, as follows:

$$\begin{cases} \text{For } b: 2 \to B & \text{do:} \quad \mathbf{P}_{\mathrm{S}}^b = \mathbf{P}_{\mathrm{S}}^b + \mathbf{P}_{\mathrm{S}}^{b-1}, \\ \text{For } b: (B-1) \to 1 & \text{do:} \quad \mathbf{P}_{\mathrm{L}}^b = \mathbf{P}_{\mathrm{L}}^b + \mathbf{P}_{\mathrm{L}}^{b+1}. \end{cases} \quad (5)$$

After the assembling, features in $\{\bar{\mathbf{F}}_1, \ldots, \bar{\mathbf{F}}_B\}$ are reconstructed by combining the short-term and long-term patterns:

$$\bar{\mathbf{F}}_b = \mathbf{P}_{\mathrm{S}}^b + \mathbf{P}_{\mathrm{L}}^b, \quad b \in \{1, \ldots, B\}. \quad (6)$$

C. Time-to-Text Semantics Extraction

LLMs are pre-trained on extensive textual data rich in semantic information, where individual words carry clear meaning. In comparison, time series data is semantically sparse, requiring an entire sequence to convey specific content. Consequently, LLMs pre-trained on textual data struggle to precisely interpret semantics present in time series data. Recent advancements in audio processing [11, 47] suggest that self-supervised learning [12] can promote models to understand the serialized audio sequences that are similar to time series data. Inspired by them, we propose the T2T module with an encoder-decoder structure to extract semantics within time series data for the LLM, as shown in Fig. 4.

Initially, the input series $\mathbf{X} \in \mathbb{R}^{H\times V}$ is divided into P patches $\{\mathbf{X}_i\}_{i=1}^P$, where $\mathbf{X}_i \in \mathbb{R}^{L\times V}$ and $L = \frac{H+O}{P}$ denotes the patch length, O is the overlap length between consecutive patches. Following [17], approximately 75% of the patches are randomly masked. Then, these patches with masks are fed into the T2T module to produce the reconstructed series $\hat{\mathbf{X}}$. To enable the T2T module to learn meaningful semantics, we train it to accurately predict the labels of time series. For a patch $\mathbf{X}_i$ (resp. $\hat{\mathbf{X}}_i$), its semantic label l_i (resp. $\hat{l}_i$) is assigned as the word with the most similar LLM text embeddings, based on the similarity $\mathbf{S}_i$:

$$\mathbf{S}_i = \text{Proj}(\mathbf{X}_i) \cdot \mathbf{E}^\top. \quad (7)$$

Here, Proj(·) linearly transforms the input to the same dimensions as the LLM text embeddings $\mathbf{E}$, and "$\top$" denotes the transpose operation.

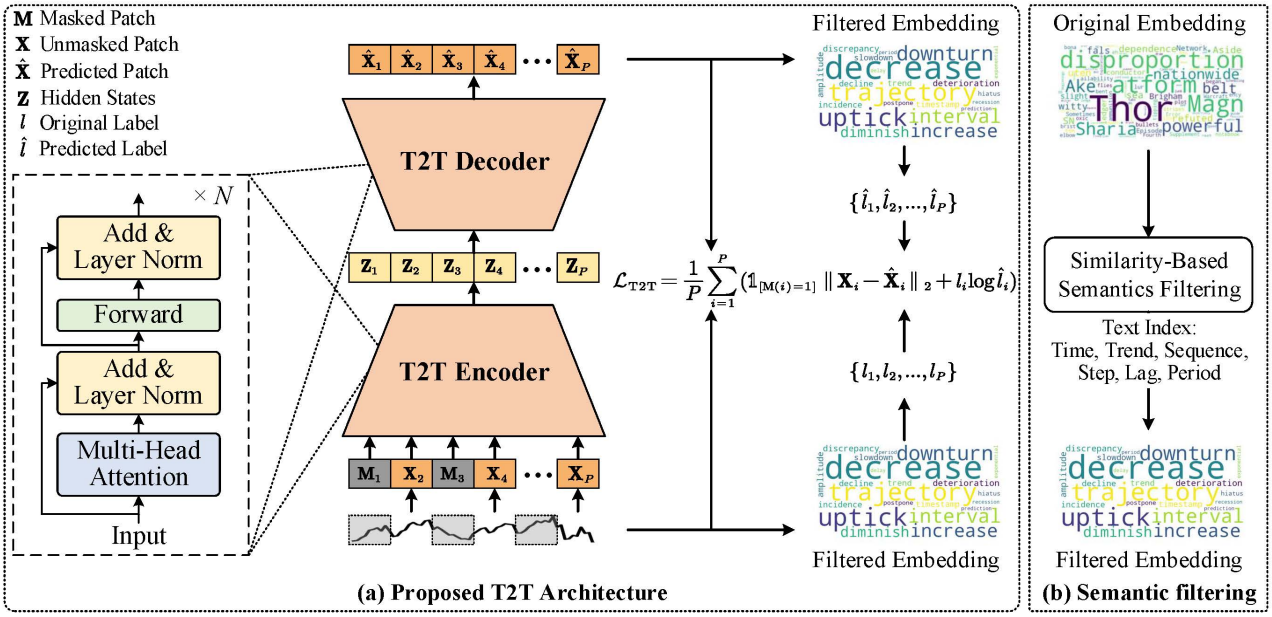


Fig. 4: The diagram of our Time-to-Text (T2T) module. The input time series is first divided into P patches, with some patches randomly masked. These patches are then fed into the encoder and decoder to facilitate T2T's accurate reconstruction and encourage T2T to learn meaningful semantics by encouraging it to accurately predict the semantic labels of the time patches.

During training, the T2T module learns precise semantic information by reconstructing the masked patches while predicting their labels. The loss function is defined as:

$$\mathcal{L}_{\text{T2T}} = \frac{1}{P}\sum_{i=1}^{P}(\mathbb{1}_{[\text{M}(i)=1]}\|\mathbf{X}_i - \hat{\mathbf{X}}_i\|_2 + l_i \log \hat{l}_i). \tag{8}$$

Here, $\mathbb{1}_{[\text{M}(i)=1]}$ is the indicator function that equals 1 if the i-th patch is masked (*i.e.*, $\text{M}(i)$=1). Further details of T2T and LLM text embeddings filtering process are provided in the **Appendix** A.

D. Efficient Training of LLM-PS

To efficiently train the LLM with huge parameters, we employ the parameter-efficient Low-Rank Adaptation (LoRA) to fine-tune the LLM on time series data. The total objective function is defined as:

$$\mathcal{L}_{\text{OBJ}} = \mathcal{L}_{\text{TIME}} + \lambda \mathcal{L}_{\text{FEAT}}, \tag{9}$$

where

$$\begin{cases} \mathcal{L}_{\text{TIME}} = \dfrac{1}{T}\displaystyle\sum_{i=1}^{T}\|\mathbf{Y}_i - \hat{\mathbf{Y}}_i\|_2, \\ \mathcal{L}_{\text{FEAT}} = \dfrac{1}{C}\displaystyle\sum_{j=1}^{C}\|\mathbf{F}_{\text{MS}}^{j} - \mathbf{F}_{\text{T2T}}^{j}\|_2. \end{cases} \tag{10}$$

Here, $\lambda > 0$ is a trade-off parameter to balance the contribution of $\mathcal{L}_{\text{TIME}}$ and $\mathcal{L}_{\text{FEAT}}$, $\mathcal{L}_{\text{TIME}}$ encourages the LLM to forecast time series $\hat{\mathbf{Y}}$ that closely matching the ground truth $\mathbf{Y}$, $\mathcal{L}_{\text{FEAT}}$ enriches the multi-scale features $\mathbf{F}_{\text{MS}}$ through semantic alignment with the T2T-generated features $\mathbf{F}_{\text{T2T}}$.

IV. Experiments

To demonstrate the effectiveness of our proposed LLM-PS, we conduct intensive experiments on multiple widely used time series datasets for various tasks, including long-term, short-term, few-shot, and zero-shot forecasting.

Baselines. We compare our LLM-PS with a large range of SOTA methods, mainly including: 1) LLM-based methods: CALF [21], GPT4TS [52], TimeCMA [20], and TimeLLM [18]; 2) Transformer-based methods: Crossformer [49], FEDformer [51], SOatten [40], PatchTST [24], iTransformer [22], ETSformer [39], and Autoformer [42]; 3) CNN-based methods: TimesNet [41], TCN [2], and MICN [36]; 4) MLP-based methods: DLinear [44], TiDE [10], and TimeMixer [37]. Besides, classical works N-HiTS [5] and N-BEATS [25] are also included for short-term forecasting.

Implementation Details. We follow [21, 52] and employ the pre-trained GPT2 model [28] (the first six layers) as the default LLM backbone. Our LLM-PS utilizes Adam as the optimizier and the learning rate is 0.0005. For the parameter-efficient LoRA, its rank, scale factor, and dropout ratio are set to 8, 32, and 0.1, respectively. Additionally, the trade-off parameter λ in Eq. (9) is set to 0.01.

A. Long-Term Forecasting

Setups. We conduct intensive experiments across popular real-world datasets, including four Electricity Transformer Temperature (ETT) subsets (ETTh1, ETTh2, ETTm1, and ETTm2), Weather, Electricity, Traffic, Illness, and ECG datasets. The input length H of time series data is set to 96, with prediction lengths T spanning $\{96, 192, 336, 720\}$. The evaluation metrics are Mean Squared Error (MSE) and

TABLE I: Average results of multivariate long-term forecasting across four different prediction lengths T in the set $\{96, 192, 336, 720\}$. The best and second best results are in **bold** and underlined, respectively. The row "1st *Count*" records the times of each method achieving the top results. Here, we reproduce the methods with superscript "⋆" according to their official codebase with the identical experimental setups as ours for fairness comparison.

Models	LLM-PS **Ours**		CALF⋆ [21]		TimeCMA [20]		TimeLLM⋆ [18]		GPT4TS⋆ [52]		SOatten [40]		PatchTST [24]		iTransformer [22]		Crossformer [49]		TimesNet [41]		MICN [36]		DLinear [44]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.354**	**0.376**	0.395	0.390	0.380	0.392	0.410	0.409	0.389	0.397	0.394	0.402	0.381	0.395	0.407	0.411	0.502	0.502	0.400	0.406	0.392	0.413	0.403	0.407
ETTm2	**0.262**	**0.314**	0.281	0.321	0.275	0.323	0.296	0.340	0.285	0.331	0.287	0.331	0.285	0.327	0.291	0.335	1.216	0.707	0.291	0.333	0.328	0.382	0.350	0.401
ETTh1	**0.418**	**0.420**	0.432	0.428	0.423	0.431	0.460	0.449	0.447	0.436	0.447	0.440	0.450	0.441	0.455	0.448	0.620	0.572	0.458	0.450	0.558	0.535	0.456	0.452
ETTh2	0.350	0.390	**0.349**	**0.382**	0.372	0.397	0.389	0.408	0.381	0.408	0.379	0.405	0.366	0.394	0.381	0.405	0.942	0.684	0.414	0.427	0.587	0.525	0.559	0.515
Weather	**0.238**	**0.269**	0.250	0.274	0.250	0.276	0.274	0.290	0.264	0.284	0.245	0.273	0.258	0.280	0.257	0.279	0.259	0.315	0.259	0.287	0.242	0.299	0.265	0.317
Electricity	**0.161**	**0.254**	0.175	0.265	0.174	0.269	0.223	0.309	0.205	0.290	0.166	0.259	0.216	0.304	0.178	0.270	0.244	0.334	0.192	0.295	0.186	0.294	0.212	0.300
Traffic	**0.427**	**0.279**	0.439	0.281	0.446	0.291	0.541	0.358	0.488	0.317	0.437	0.286	0.555	0.361	0.428	0.282	0.550	0.304	0.620	0.336	0.541	0.315	0.625	0.383
ILI	**1.735**	0.854	1.861	0.924	1.922	0.921	1.829	0.924	1.871	**0.852**	1.868	0.930	2.145	0.897	2.258	0.957	3.749	1.284	2.267	0.927	2.985	1.186	4.453	1.553
ECG	**0.225**	**0.250**	0.258	0.260	0.263	0.285	0.250	0.264	0.262	0.260	0.264	0.279	0.253	0.277	0.257	0.271	0.244	0.269	0.291	0.305	0.305	0.314	0.291	0.307
1st *Count*	**15**		2		0		0		1		0		0		0		0		0		0		0	

TABLE II: Results of short-term forecasting across monthly, quarterly, yearly, and others subsets.

Models		LLM-PS (**Ours**)	CALF [21]	TimeLLM [18]	GPT4TS [52]	PatchTST [24]	iTransformer [22]	ETSformer [39]	FEDformer [51]	Autoformer [42]	TimesNet [41]	TCN [2]	N-HiTS [5]	N-BEATS [25]	DLinear [44]	LSSL [14]	LSTM [16]
Yearly	SMAPE	**13.277**	13.314	13.419	13.531	13.477	14.252	18.009	13.728	13.974	13.387	14.920	13.418	13.436	16.965	16.675	176.040
	MASE	**2.973**	3.009	3.005	3.015	3.019	3.208	4.487	3.048	3.134	2.996	3.364	3.045	3.043	4.283	19.953	31.033
	OWA	**0.780**	0.786	0.789	0.793	0.792	0.840	1.115	0.803	0.822	0.786	0.880	0.793	0.794	1.058	4.397	9.290
Quarterly	SMAPE	**9.995**	10.049	10.110	10.177	10.380	10.755	13.376	10.792	11.338	10.100	11.122	10.202	10.124	12.145	65.999	172.808
	MASE	**1.164**	1.166	1.178	1.194	1.233	1.284	1.906	1.283	1.365	1.182	1.360	1.194	1.169	1.520	17.662	19.753
	OWA	0.878	**0.871**	0.889	0.898	0.921	0.957	1.302	0.958	1.012	0.890	1.001	0.899	0.886	1.106	9.436	15.049
Monthly	SMAPE	**12.585**	12.624	12.980	12.894	12.959	13.721	14.588	14.260	13.958	12.679	15.626	12.791	12.677	13.514	64.664	143.237
	MASE	**0.924**	0.922	0.963	0.956	0.970	1.074	1.368	1.102	1.103	0.933	1.274	0.969	0.937	1.037	16.245	16.551
	OWA	**0.871**	**0.871**	0.903	0.897	0.905	0.981	1.149	1.012	1.002	0.878	1.141	0.899	0.880	0.956	9.879	12.747
Others	SMAPE	**4.550**	4.773	4.795	4.940	4.952	5.615	7.267	4.954	5.485	4.891	7.186	5.061	4.925	6.709	121.844	186.282
	MASE	**3.089**	3.119	3.178	3.228	3.347	3.977	5.240	3.264	3.865	3.302	4.677	3.216	3.391	4.953	91.650	119.294
	OWA	**0.966**	0.990	1.006	1.029	1.049	1.218	1.591	1.036	1.187	1.035	1.494	1.040	1.053	1.487	27.273	38.411
1st *Count*		**11**	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TABLE III: Average results of few-shot forecasting with 10% training data, the prediction lengths $T \in \{96, 192, 336, 720\}$.

Models	LLM-PS **Ours**		CALF [21]		TimeLLM [18]		GPT4TS [52]		PatchTST [24]		Crossformer [49]		FEDformer [51]		TimesNet [41]		MICN [36]		DLinear [44]		TiDE [10]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.497**	**0.454**	0.504	0.462	0.636	0.512	0.608	0.500	0.557	0.483	1.340	0.848	0.696	0.572	0.673	0.534	0.970	0.674	0.567	0.499	0.515	0.469
ETTm2	**0.281**	**0.324**	0.302	0.330	0.348	0.343	0.303	0.336	0.295	0.334	1.985	1.048	0.356	0.392	0.321	0.354	1.073	0.716	0.329	0.382	0.303	0.337
ETTh1	**0.632**	0.546	0.644	**0.541**	0.765	0.584	0.689	0.555	0.683	0.546	1.744	0.914	0.750	0.607	0.865	0.625	1.405	0.814	0.647	0.552	0.779	0.604
ETTh2	**0.409**	**0.420**	0.419	0.427	0.589	0.498	0.579	0.497	0.550	0.487	3.139	1.378	0.553	0.525	0.476	0.463	2.533	1.158	0.441	0.458	0.421	0.428
1st *Count*	**7**		1		0		0		0		0		0		0		0		0		0	

TABLE IV: Average results of zero-shot forecasting, where prediction lengths $T \in \{96, 192, 336, 720\}$. The term "ETTh1 → ETTm1" indicates that models trained on ETTh1 and are evaluated on ETTm1, the convention is also followed for other terms.

Models	LLM-PS **Ours**		CALF [21]		TimeLLM [18]		GPT4TS [52]		PatchTST [24]		Crossformer [49]		FEDformer [51]		TimesNet [41]		MICN [36]		DLinear [44]		TiDE [10]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1 → ETTm1	**0.721**	**0.541**	0.755	0.574	0.847	0.565	0.798	0.574	0.894	0.610	0.999	0.736	0.765	0.588	0.794	0.575	1.439	0.780	0.760	0.577	0.774	0.574
ETTh1 → ETTm2	0.316	0.361	0.316	**0.355**	0.315	0.357	0.317	0.359	0.318	0.362	1.120	0.789	0.357	0.403	0.339	0.370	2.428	1.236	0.399	0.439	**0.314**	**0.355**
ETTh2 → ETTm1	**0.714**	**0.552**	0.836	0.586	0.868	0.595	0.920	0.610	0.871	0.596	1.195	0.711	0.741	0.588	1.286	0.705	0.764	0.601	0.778	0.594	0.841	0.590
ETTh2 → ETTm2	0.322	**0.359**	**0.319**	0.360	0.322	0.363	0.331	0.371	0.420	0.433	2.043	1.124	0.365	0.405	0.361	0.390	0.527	0.519	0.496	0.496	0.321	0.364
1st *Count*	**5**		2		0		0		0		0		0		0		0		0		2	

Mean Absolute Error (MAE), where lower values of these metrics indicate better model performance.

Results. Tab. I reports the brief average results of long-term forecasting with various prediction lengths. Firstly, we can observe that our LLM-PS surpasses the baseline methods in most instances, achieving the top results in 15 of 18 cases. Secondly, compared with the SOTA LLM-based methods, *i.e.*, CALF, TimeLLM, and GPT4TS, our approach achieves consistent MSE/MAE reductions of 6%/3%, 11%/9%, and 9%/5%, respectively. Thirdly, our method significantly outperforms traditional deep learning methods based on Transformer, CNN, and MLP, especially on the Traffic, ILI, and ECG datasets. These results indicate that our LLM-PS can precisely predict long-term time series by effectively leveraging temporal patterns and semantics within the input series.

B. Short-Term Forecasting

Setups. We evaluate the M4 dataset [23], which records marketing data in monthly, quarterly, yearly, and others. For short-term forecasting, the prediction horizons are relatively small and range in $[6, 48]$, while the input lengths are twice their corresponding prediction horizons. The evaluation metrics are Mean Absolute Scaled Error (MSAE), Symmetric Mean Absolute Percentage Error (SMAPE), and Overall Weighted Average (OWA).

Results. The results of short-term forecasting across monthly, quarterly, yearly, and others subsets are shown in Tab. II. Our proposed LLM-PS achieves the top results in 11 of 12 cases.

C. Few/Zero-Shot Forecasting

Setups. LLMs possess powerful few-shot and zero-shot learning capabilities [18, 52], which are crucial for time series forecasting in real-world scenarios. We also verify the few-shot and zero-shot learning performance of our LLM-PS. For few-shot forecasting, there are only 10% training data of ETT datasets are available. For zero-shot forecasting, LLMs trained on one dataset are directly tested on other datasets. The training setups are the same as those in long-term forecasting.

Results. The average results over various prediction horizons for the challenging few-shot forecasting tasks are reported in Tab. III. In such cases, our LLM-PS always achieves the best results in 7 of 8 cases, which demonstrates that our LLM-PS can effectively learn valuable temporal information only using limited data. Compared with the LLM-based methods, *i.e.*, CALF, TimeLLM, and GPT4TS, our method yields 3%, 17%, and 13% performance improvements, respectively.

In addition to the few-shot learning tasks, we also considered the more challenging zero-shot learning tasks. The results are provided in Tab. IV. Our method significantly outperforms the comparison methods in 5/8 cases. Notably, our LLM-PS outperforms CALF, TimeLLM, and GPT4TS, which also fine-tune the LLM for time series forecasting, showing performance improvements of 5%, 8%, and 9%, respectively.

D. Model Analysis

Multi-Scale Feature Extraction. We compare our MSCNN against existing multi-scale features extraction techniques [19], which convert the input series into diverse scales using average or max pooling layers with varying window sizes. As shown in Fig. 5a, our method consistently outperforms those employing pooling operations. These results indicate that our method excels in extracting the multi-scale features, thereby effectively capturing the short-term and long-term patterns in time series.

Temporal Patterns Decoupling. We compare our wavelet-transform-based decoupling with the Fourier-transform-based and pooling-based decoupling techniques used in existing methods [37, 42]. As shown in Fig. 5b, our decoupling operation based on wavelet transform achieves better performance than those based on Fourier transform and average pooling. The short-term and long-term components decomposed by our LLM-PS and compared methods are visualized in Fig. 6.

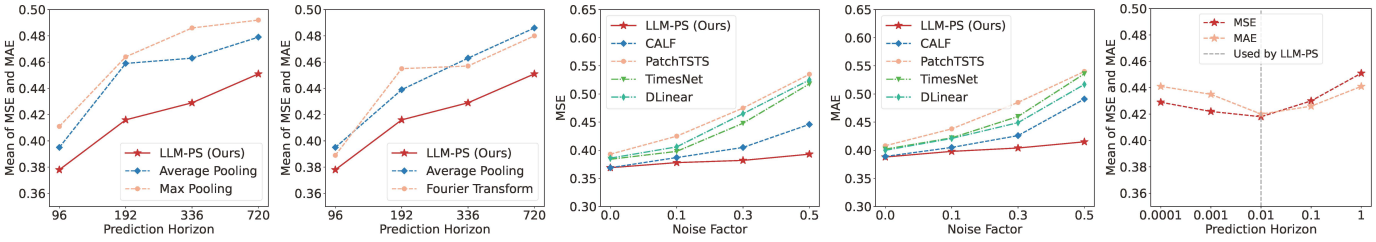
Semantic Information Utilization. Tab. V reports the results of ablating our proposed T2T module for semantic information learning. It can be found that the performance of LLM decreases in 9 of 10 cases, which demonstrates that the semantic information extracted by our T2T is helpful in enhancing the TSF performance of LLM.

TABLE V: Long-term forecasting results of ablating our proposed T2T module on the ETTh1 dataset.

Type	MSE / MAE				
	96	192	336	720	Mean
w/o T2T	0.373 / 0.395	**0.416** / 0.425	0.439 / 0.432	0.464 /0.470	0.426 /0.431
LLM-PS	**0.369** / **0.388**	0.418 / **0.415**	**0.432** / **0.426**	**0.452** / **0.451**	**0.418** / **0.420**

Model Efficiency. We evaluate the efficiency of our method with other LLM fine-tuning methods on four datasets, including ETTh1, ETTm1, Weather, and Traffic. For a fair comparison, all experiments adopt the same experimental settings and LLM backbone (*i.e.*, GPT2), where input length, prediction length, and patience for early stopping are set to 96, 96, and 5, respectively. All experiments are conducted on a single NVIDIA RTX 4090 GPU. As reported in Tab. VI, our proposed LLM-PS achieves the best performance with significantly lower training costs. Compared with LLMMixer, which also learns multi-scale features from the multiple input series with various scales, our method attains a 17% performance edge while utilizing just 9% of the training time. These experimental results indicate that our method can efficiently and effectively learn the temporal patterns and semantic information from time series data, and thus achieving reliable TSF performance.

Noisy Data. Time series data in real-world applications is usually noisy due to measurement errors and missing values. In such cases, the target time series is more challenging than that in training data, and the models are hard to predict reliably. To assess the robustness of our LLM-PS against noise, we evaluate it on the ETTh1 dataset with Gaussian noise, where the noise factors are in $[0.0, 0.1, 0.3, 0.5]$. Both the input and prediction lengths are set to 96. As reported



(a) (b) (c) (d) (e)

Fig. 5: Analysis of (a) multi-scale feature extraction and (b) temporal patterns decoupling. Subfigures (c) and (d) show the MSE/MAE of various methods on noisy ETTh1 datasets. Subfigure (e) analyzes parametric sensitivities of λ in Eq. (9).

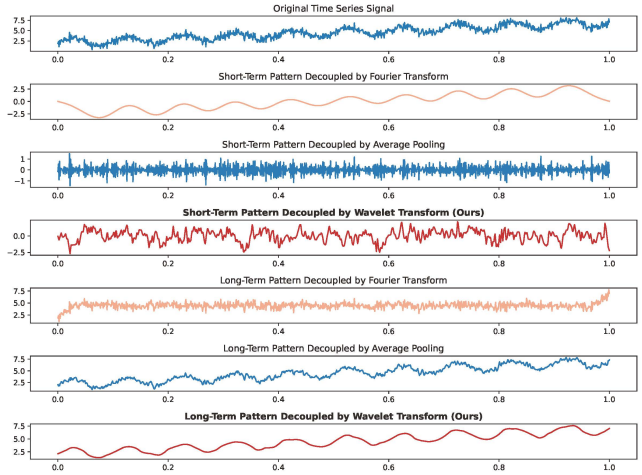


Fig. 6: Visualization results of temporal patterns decoupled by our proposed wavelet-transform-based decoupling technique and other widely-used decoupling methods based on Fourier transform and average pooling.

TABLE VI: The fine-tuning costs, mean MSE/MAE across four datasets of our LLM-PS and other LLM-based methods.

Model	Time (s)				Mean MSE	Mean MAE
	ETTh1	ETTm1	Weather	Traffic		
GPT4TS [52]	421	1140	4565	59164	0.339	0.323
Time-LLM [18]	2780	11929	36188	465136	0.372	0.346
LLMMixer [19]	635	2493	9640	10464	0.372	0.346
CALF [21]	354	1394	1259	4929	0.315	0.302
LLM-PS (Ours)	**192**	**481**	**260**	**1092**	**0.301**	**0.298**

in Fig. 5c&5d, our method consistently achieves superior performance across various noise factors. In particular, our method outperforms other comparison approaches by an even more significant margin as the noise factor increases. These experimental results demonstrate that our LLM-PS is robust to noise and can be effectively applied to real-world scenarios for time series forecasting.

Parameter Sensitivity Analysis Our LLM-PS has one tuning parameter, denoted as λ. We analyze its sensitivities on the ETTh1 dataset by varying λ within the range of $\{0.0001, 0.001, 0.01, 0.1, 1\}$ and observe the mean MSE/MAE across predict lengths spanning $\{96, 192, 336, 720\}$, as shown in Fig. 5e. Despite the large fluctuations in λ, the MSE/MAE curve of our LLM-PS remains relatively stable. These results demonstrate the robustness of our LLM-PS against parameter variations. Furthermore, our LLM-PS achieves its best performance when $\lambda = 0.01$, so we adopt this parameter configuration in our method.

V. Conclusion

In this paper, we identify the intrinsic characteristics of time series data, *i.e.*, diverse temporal patterns and semantic sparsity. These properties are critical for reliable time series forecasting but are usually neglected by existing LLM-based methods, and thus resulting in suboptimal performance. To address this problem, we propose LLM-PS, a novel TSF framework that learns fundamental temporal patterns and valuable semantics from time series through the novel MSCNN and T2T modules. As a result, our LLM-PS can comprehensively understand time series data, thereby enabling accurate generation of time series. Our intensive experiments demonstrate that LLM-PS achieves SOTA performance across multiple benchmark datasets spanning critical real-world domains, mainly including finance, energy, transportation, and healthcare.

Acknowledgments

C. Gong's research is supported by NSF of China (Nos: 62336003, 12371510), and D. Tao's research is partially supported by NTU RSR and Start Up Grants.

References

[1] R. A. Angryk, P. C. Martens, B. Aydin, D. Kempton, S. S. Mahajan, S. Basodi, A. Ahmadzadeh, X. Cai, S. Filali Boubrahimi, S. M. Hamdi *et al.*, "Multivariate time series dataset for space weather data analytics," *Scientific Data*, vol. 7, no. 1, p. 227, 2020.

[2] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.

[4] D. Cao, F. Jia, S. O. Arik, T. Pfister, Y. Zheng, W. Ye, and Y. Liu, "Tempo: Prompt-based generative pre-trained

transformer for time series forecasting," in *International Conference on Learning Representations (ICLR)*.

[5] C. Challu, K. G. Olivares, B. N. Oreshkin, F. Garza, M. Mergenthaler, and A. Dubrawski, "N-HiTs: Neural hierarchical interpolation for time series forecasting," *arXiv:2201.12886*, 2022.

[6] M. Cheng, Q. Liu, Z. Liu, H. Zhang, R. Zhang, and E. Chen, "Timemae: Self-supervised representations of time series with decoupled masked autoencoders," *arXiv:2303.00320*, 2023.

[7] H. Chung, J. Kim, J.-m. Kwon, K.-H. Jeon, M. S. Lee, and E. Choi, "Text-to-ecg: 12-lead electrocardiogram synthesis conditioned on clinical text reports," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[8] R. B. Cleveland, W. S. Cleveland, J. E. McRae, I. Terpenning *et al.*, "Stl: A seasonal-trend decomposition," *J. off. Stat*, vol. 6, no. 1, pp. 3–73, 1990.

[9] W. T. Cochran, J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling, D. E. Nelson, C. M. Rader, and P. D. Welch, "What is the fast fourier transform?" *Proceedings of the IEEE*, vol. 55, no. 10, pp. 1664–1674, 1967.

[10] A. Das, W. Kong, A. Leach, R. Sen, and R. Yu, "Long-term forecasting with TiDE: Time-series dense encoder," *arXiv:2304.08424*, 2023.

[11] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *Transactions on Machine Learning Research (TMLR)*, 2023.

[12] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[13] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 43, no. 2, pp. 652–662, 2019.

[14] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *International Conference on Learning Representations (ICLR)*, 2022.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput (NC)*, 1997.

[17] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLR)*, vol. 29, pp. 3451–3460, 2021.

[18] M. Jin, S. Wang, L. Ma, Z. Chu, J. 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," in *International Conference on Learning Representations (ICLR)*, 2024.

[19] M. Kowsher, M. S. I. Sobuj, N. J. Prottasha, E. A. Alanis, O. O. Garibay, and N. Yousefi, "Llm-mixer: Multiscale mixing in llms for time series forecasting," *arXiv:2410.11674*, 2024.

[20] C. Liu, Q. Xu, H. Miao, S. Yang, L. Zhang, C. Long, Z. Li, and R. Zhao, "Timecma: Towards llm-empowered multivariate time series forecasting via cross-modality alignment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 18, 2025, pp. 18 780–18 788.

[21] 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," *arXiv:2403.07300*, 2024.

[22] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "iTransformer: Inverted transformers are effective for time series forecasting," *International Conference on Learning Representations (ICLR)*, 2024.

[23] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: Results, findings, conclusion and way forward," *International Journal of Forecasting (IJF)*, vol. 34, no. 4, pp. 802–808, 2018.

[24] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *International Conference on Learning Representations (ICLR)*, 2023.

[25] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," *International Conference on Learning Representations (ICLR)*, 2019.

[26] N. Oreshkin, Boris N and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv:1905.10437*, 2019.

[27] A. Patton, "Copula methods for forecasting multivariate time series," *Handbook of economic forecasting*, vol. 2, pp. 899–960, 2013.

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[29] P. K. Rubenstein, C. Asawaroengchai, D. D. Nguyen, A. Bapna, Z. Borsos, F. d. C. Quitry, P. Chen, D. E. Badawy, W. Han, E. Kharitonov *et al.*, "Audiopalm: A large language model that can speak and listen," *arXiv:2306.12925*, 2023.

[30] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting (IJF)*, vol. 36, no. 3, pp. 1181–1191, 2020.

[31] M. Tan, M. Merrill, V. Gupta, T. Althoff, and T. Hartvigsen, "Are language models actually useful for time series forecasting?" *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, pp. 60 162–60 191, 2024.

[32] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[33] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv:2302.13971*, 2023.

[34] A. Trindade, "ElectricityLoadDiagrams20112014," UCI Machine Learning Repository, 2015, DOI: https://doi.org/10.24432/C58C86.

[35] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[36] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao, "MICN: Multi-scale local and global context modeling for long-term series forecasting," in *International Conference on Learning Representations (ICLR)*, 2022.

[37] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J. Y. Zhang, and J. ZHOU, "Timemixer: Decomposable multiscale mixing for time series forecasting," in *International Conference on Learning Representations (ICLR)*, 2024.

[38] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: a survey," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, 2023, pp. 6778–6786.

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

[40] H. Wu, "Revisiting attention for multivariate time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 20, 2025, pp. 21 528–21 535.

[41] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "TimesNet: Temporal 2d-variation modeling for general time series analysis," in *International Conference on Learning Representations (ICLR)*, 2023.

[42] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 22 419–22 430, 2021.

[43] H. Wu, H. Zhou, M. Long, and J. Wang, "Interpretable weather forecasting for worldwide stations with a unified deep model," *Nature Machine Intelligence (NMI)*, vol. 5, no. 6, pp. 602–611, 2023.

[44] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 11 121–11 128.

[45] D. Zhang, "Wavelet transform," *Fundamentals of Image Data Mining: Analysis, Features, Classification and Retrieval*, pp. 35–44, 2019.

[46] G. P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *European Journal of Operational Research (EJOR)*, vol. 160, no. 2, pp. 501–514, 2005.

[47] X. Zhang, D. Zhang, S. Li, Y. Zhou, and X. Qiu, "Speechtokenizer: Unified speech tokenizer for speech language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[48] X. Zhang, R. R. Chowdhury, R. K. Gupta, and J. Shang, "Large language models for time series: A survey," *arXiv:2402.01801*, 2024.

[49] Y. J. Zhang, Yunhao, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *International Conference on Learning Representations (ICLR)*, 2023.

[50] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[51] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International Conference on Machine Learning (ICML)*. PMLR, 2022, pp. 27 268–27 286.

[52] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One Fits All: Power general time series analysis by pretrained lm," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023.

APPENDIX

A. Time-to-Text Module

Model Configuration. In the T2T module, the encoder and decoder contain 4 and 1 transformer layer, respectively. Each transformer layer consists of the multi-head attention layer, feedforward network, and layer normalization. The patch size, hidden dimension, feedforward hidden size, and output size are set to 24, 96, 384, and 24, respectively.

Training Setups. During training, the T2T module is optimized by Adam, where the learning rate is set to 0.001. The masking ratio of input time series patches is set to 0.75.

Semantic Filtering. In the LLM word embeddings, there are numerous inrelevant semantic information for Time Series Forecasting (TSF). As shown in Fig. 4, the original word embeddings $\mathbf{E}_{\text{ori}} \in \mathbb{R}^{W \times D}$ (where W and D are the length and dimension of $\mathbf{E}_{\text{ori}}$) of the GPT2 model includes embedding with corresponding word such as "Thro", "Magn", and "belt", which are irrelevant to TSF. To facilitate T2T in extracting valuable semantic information for TSF, we filter the LLM word embeddings based on text indices relevant to TSF.

Specifically, given the text indices $\{t_i\}_i^I$ (I denotes the number of text indices) corresponding to the TSF, the similarities between their word embeddings $\hat{\mathbf{E}} = \{\mathbf{E}_{\text{ori}}[t_i]\}_{i=1}^I$ and LLM word embeddings $\mathbf{E}_{\text{ori}}$ are computed as follows:

$$\mathbf{s} = \frac{\hat{\mathbf{E}} \cdot \mathbf{E}_{\text{ori}}}{\|\hat{\mathbf{E}}\|_2 \|\mathbf{E}_{\text{ori}}\|_2}. \tag{11}$$

Then, we select the top 100 most similar word embeddings as the final word embeddings $\mathbf{E}$ for T2T training.