



Article

ADTime: Adaptive Multivariate Time Series Forecasting Using LLMs

Jinglei Pei ¹, Yang Zhang ², Ting Liu ¹ , Jingbin Yang ¹, Qinghua Wu ¹ and Kang Qin ^{2,*}

¹ Institute of Computing Technology, University of Chinese Academy of Sciences, Beijing 100190, China; peijinglei24s@ict.ac.cn (J.P.); liuting19g@ict.ac.cn (T.L.); yangjingbin20g@ict.ac.cn (J.Y.); wuqinghua@ict.ac.cn (Q.W.)

² Research Institute of Petroleum Processing, SINOPEC, Beijing 100083, China; zhangyang.ripp@sinopec.com

* Correspondence: qinkang.ripp@sinopec.com;

Abstract: Large language models (LLMs) have recently demonstrated notable performance, particularly in addressing the challenge of extensive data requirements when training traditional forecasting models. However, these methods encounter significant challenges when applied to high-dimensional and domain-specific datasets. These challenges primarily arise from inability to effectively model inter-variable dependencies and capture variable-specific characteristics, leading to suboptimal performance in complex forecasting scenarios. To address these limitations, we propose **ADTime**, an adaptive LLM-based approach for multivariate time series forecasting. ADTime employs advanced preprocessing techniques to identify latent relationships among key variables and temporal features. Additionally, it integrates temporal alignment mechanisms and prompt-based strategies to enhance the semantic understanding of forecasting tasks by LLMs. Experimental results show that ADTime outperforms state-of-the-art methods, reducing MSE by 9.5% and MAE by 6.1% on public datasets, and by 17.1% and 13.5% on domain-specific datasets. Furthermore, zero-shot experiments on real-world refinery datasets demonstrate that ADTime exhibits stronger generalization capabilities across various transfer scenarios. These findings highlight the potential of ADTime in advancing complex, domain-specific time series forecasting tasks.



Academic Editor: Laura Po

Received: 19 February 2025

Revised: 9 April 2025

Accepted: 11 April 2025

Published: 17 April 2025

Citation: Pei, J.; Zhang, Y.; Liu, T.; Yang, J.; Wu, Q.; Qin, K. ADTime: Adaptive Multivariate Time Series Forecasting Using LLMs. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 35. <https://doi.org/10.3390/make7020035>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: time series; time series forecasting; large language models

1. Introduction

Time series forecasting is widely used in various domains, including weather prediction [1], health monitoring [2], energy management [3,4], and financial forecasting [5]. These methods are essential for analyzing temporal trends, patterns, and relationships among variables, providing critical data support for economic and infrastructure planning.

Benefiting from self-attention mechanisms, transformer-based approaches have leveraged strengths in modeling long-term dependencies for sequential data. They enhance models' ability to capture temporal patterns by modifying the model architecture or employing processing techniques for time series data. However, the promising performance of transformer-based methods relies heavily on large and domain-specific pretraining datasets. Traditional forecasting approaches generally require domain-specific models, which demand significant expertise and have limited applicability across domains. In contrast, the development of pretrained large language models (LLMs) such as GPT [6,7] and LLaMA [8] offers a promising alternative. These models acquire general domain knowledge through pretraining on vast amounts of textual data, enabling their application across a wide range

of downstream tasks [9]. Additionally, LLMs exhibit exceptional performance in few-shot and zero-shot learning, leading to their increasing adoption in time series forecasting even in data-scarce scenarios. Therefore, many studies have directly fed time series data into LLMs. Through techniques such as prompt engineering, LLMs can surpass domain-specific models with minimal or no reliance on downstream training data.

Real-world time series data often have high dimensionality and exhibit significant variations across different domains, posing several challenges for domain-specific time series forecasting with large models. First, real-world time series data often exhibit unique temporal features that cannot be effectively captured through a single fixed strategy. Existing approaches tend to apply the same strategy across all data dimensions; for instance, Tempo performs temporal decomposition uniformly across all variables, which may lead to information loss and interference. Second, existing LLM-based forecasting methods typically model each univariate time series independently using a channel-independent approach, which neglects inter-variable dependencies [10,11]. For example, in PatchTST, each model's input token contains information from only a single channel.

To address these challenges, we propose an adaptive time series forecasting framework called **ADTime**, which is designed to efficiently leverage inter-variable dependencies and diverse temporal patterns in domain-specific time series analysis. To enable flexible time series feature learning, we design an adaptive time series processing module that dynamically adjusts processing strategies based on data features. After performing correlation-based joint modeling of variables and analyzing each channel's periodicity using FFT, STL decomposition is applied to strongly periodic data. On the other hand, weakly periodic data retain their original sequences, ensuring effective feature extraction and adaptation to diverse temporal patterns. To enhance the ability of LLMs to understand temporal features, we integrate temporal text alignment and prompt engineering into the design of the embedding layers within the LLM. Extensive evaluations show that ADTime offers an effective solution for diverse domain-specific datasets and that it excels in both few-shot and zero-shot time series forecasting scenarios. The main contributions of this work are as follows:

- We investigate the domain-specific correlations among multivariate data and the heterogeneous temporal characteristics across different channels. This leads to the development of an elastic forecasting method tailored to diverse temporal patterns, enhancing the capacity of LLMs to process multi-scale temporal information.
- We propose an adaptive LLM-based framework called ADTime, which integrates two modules in order to process multivariate time series data. First, we apply channel clustering and temporal decomposition based on the distinct temporal features of each channel. This is followed by an embedding construction module that uses text alignment and adaptively designed prompts to improve the model's ability to capture temporal patterns.
- Experimental results on seven forecasting benchmarks show that our model outperforms existing methods. Specifically, ADTime reduces MSE by 9.5% and MAE by 6.1% on public datasets, while achieving reductions of 17.1% and 13.5% on real-world refinery datasets (the refinery dataset utilized in this study was provided by Sinopec, and contains sensor and product generation data from multiple refineries that are not publicly available due to privacy concerns). Additionally, through ablation studies and model analysis, we further investigate the key factors contributing to ADTime's effectiveness.

2. Related Work

This paper focuses on the development of multivariate time series forecasting. In this section, we provide a comprehensive review of key developments in time series decomposition, channel strategies, and the application of transformer-based models, highlighting both their contributions and limitations.

2.1. Transformer-Based Time Series Models

In recent years, transformer-based models [12–15] have been widely applied to time series tasks. Informer [16] introduced the ProbSparse self-attention mechanism to distill input sequences, resulting in enhanced efficiency when processing long sequences. Autoformer [17] is an auto-correlation mechanism for uncovering dependencies between variables, while FEDformer [18] leverages a Fourier-enhanced block and Wavelet-enhanced block to capture key structures in time series data. PatchTST [10] turns time series into patches in order to capture comprehensive semantic information, while Pathformer [19] introduces a dual attention mechanism for enhanced transformer-based time series modeling capabilities. However, these approaches are not suitable for domain-specific time series prediction with small amounts of data.

Meanwhile, the development of pretrained large language models (LLMs) has provided new possibilities for time series analysis [20]. The robust text generation and pattern recognition capabilities of LLMs provide a foundation for their application to time series tasks. LLM-centered approaches primarily enhance model performance through two main strategies. One common approach is to fine-tune LLMs for specific downstream tasks [21]. For example, GPT4TS [22] employs a frozen pretrained transformer in which the self-attention and feedforward layers of the LLM remain frozen while fine-tuning is performed for time series tasks. While noteworthy, this method still requires training on task-specific data.

Another strategy is to modify the input format for LLMs, allowing them to be used directly without further training. Several studies [23] have adopted multimodal processing techniques that align time series features with LLM representations. For instance, TEST [24] applies contrastive learning to generate embeddings for time series tokens, while TIME-LLM [25] employs multihead attention to reprogram time series data into textual representations. Additionally, several works have leveraged prompts to improve performance on specific tasks. For example, TEMPO [26] utilizes soft prompt strategies to accommodate the dynamic nature of time series data. However, these approaches all apply the uniform embedding construction method across all variants, which fails to capture the unique characteristics of time series data in different domains. As a result, their performance in real-world temporal tasks remains limited.

2.2. Decomposition of Time Series

Time series decomposition is a classic method in time series analysis [18,26,27] that involves breaking down a time series into multiple independent components such as trend, seasonality, and residuals [28]. For instance, the trend component represents the long-term change in the time series, while the seasonality component captures shorter-term periodic fluctuations. N-BEATS [29] addresses decomposition by incorporating a neural network-based decomposition module to estimate local decomposition components, while Autoformer [17] applies traditional mean filtering to convolve the time series, then extracts the trend component while treating the remainder as the seasonality component. TimesNet [30] leverages Fourier transformation to decompose time series in the frequency domain, extracting multiple seasonal components with varying periodic lengths in order to better capture diverse periodic patterns.

2.3. Channel Strategies in Time Series Models

In multivariate time series forecasting, channel strategies significantly influence model performance. These strategies can be broadly classified into two categories, namely, channel-independent (CI) and channel-dependent (CD), based on how the inter-channel dependencies are modeled.

CI strategies treat multivariate time series as N independent univariate series, then predict each channel separately [10,11]. CI strategies excel at capturing the temporal dynamics of individual variables and may yield higher prediction accuracy, particularly when inter-channel similarities are low; however, they often overlook potential interactions between channels, which can limit their generality and robustness.

On the other hand, CD strategies treat all channel data as joint inputs while leveraging cross-channel information to model complex interdependencies [16–18,31]. TimesNet [30] uses CNNs to capture cross-dimensional dependencies, while Crossformer [32] employs a two-stage attention mechanism to effectively model both cross-time and cross-dimensional dependencies. In general, CD strategies aim to learn and represent intricate relationships among variables; however, they can be vulnerable to noise in time series data. Transformer-based models relying on CD strategies are particularly prone to overfitting noise, which can degrade their performance [33]. Therefore, designing an effective channel strategy is crucial for realizing improved forecasting accuracy [34,35].

3. Methodology

A multivariate time series can be represented as $X = [X_1, \dots, X_K] \in R^{K \times N}$, where K denotes the length of the historical data, N denotes the number of variables observed, and $X_t \in R^N$ denotes the observation value at time t . Specifically, X_t is composed of individual components x_t^i , where x_t^i denotes the observation value of the i -th channel at time t . Thus, and we can write $X_t = x_t^i, i = 1, \dots, N$.

The goal of multivariate time series forecasting is to learn a function that accurately predicts the time series data for the next H time steps based on the historical series of K time steps. In this context, the predicted values are denoted by $\hat{X}_t$, which represents the forecasted observation vector at time t :

$$Y = [\hat{X}_t, \dots, \hat{X}_{t+H-1}] = F(X_{t-K}, \dots, X_{t-1})$$

where Y is the sequence of predicted values over the next H time steps and $F(\cdot)$ is the forecasting function.

As depicted in Figure 1, our proposed framework comprises three core components: (1) multivariate time series processing, (2) temporal alignment mechanisms, and (3) adaptive prompting for LLMs. The framework begins with the temporal processing module, which first identifies inter-variable dependencies using the channel clustering module. It then evaluates the periodicity of each channel to determine whether STL decomposition is required. Following this, the aligned series are mapped to the textual semantic space. Ultimately, adaptive prompts are integrated to enhance the temporal understanding capabilities of LLMs, significantly improving their overall forecasting performance.

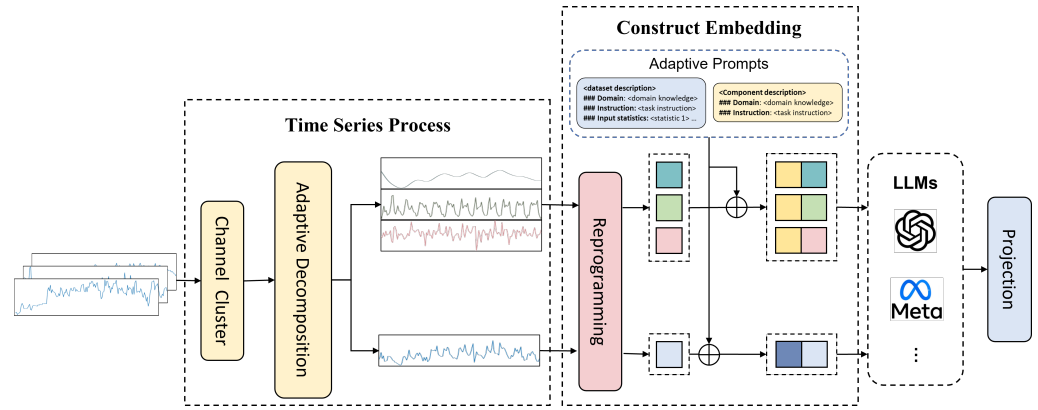


Figure 1. The model framework of ADTime.

During the training process, only the parameters of the embedding layer and output projection layer need to be updated, while the backbone LLM remains frozen. This approach significantly enhances efficiency and reduces resource requirements compared to building or fine-tuning an entire time series model.

3.1. Multivariate Time Series Process

In order to address the inherent complexities and varied characteristics of multivariate domain data, we apply a set of strategies consisting of channel clustering and time series decomposition. These strategies can work together to improve the robustness and predictive performance of time series models, especially in the presence of complex dependencies and varying periodic behaviors across channels.

Channel Clustering. In multivariate time series analysis, the correlation between different dimensions is critical for predictive performance. When using domain-specific data, the interrelationships between features reflect underlying mechanisms and system parameters. Highly correlated dimensions often exhibit similar time series patterns, whereas low-correlation dimensions reveal more independent variation characteristics [35]. For channels with high similarity, the individual channel identity information contributes minimally to predictive power, as the model can derive comparable insights from other similar channels. By leveraging this channel similarity, reliance on specific identity information can be minimized, allowing the model to focus on shared features across channels for improved generalization and predictive performance.

To combine the strengths of both independent and dependent approaches, we propose a channel clustering strategy. As illustrated in Figure 2, the channel clustering module captures essential cross-channel dependencies by clustering channels based on their feature correlations. We use the distance correlation to measure interdependencies between channels and cluster time series when the distance correlation exceeds a predefined threshold Th_{cor} .

After channel clustering, the channels are grouped into C clusters. Each cluster i contains c_i channels and can be denoted as $\tilde{X}^i \in \mathbb{R}^{K \times c_i}$, $\{i = 1, \dots, C\}$. Channels with high similarity undergo joint embedding, enabling the model to exploit intrinsic connections and realize enhanced predictive accuracy. For other variants, we apply a channel-independent strategy that allows these channels pass through the projection layer independently, which reduces the influence of irrelevant data and improves the model's generalization ability. The output of the channel clustering module is represented as $\tilde{X}^i \in \mathbb{R}^K$, $\{i = 1, \dots, C\}$.

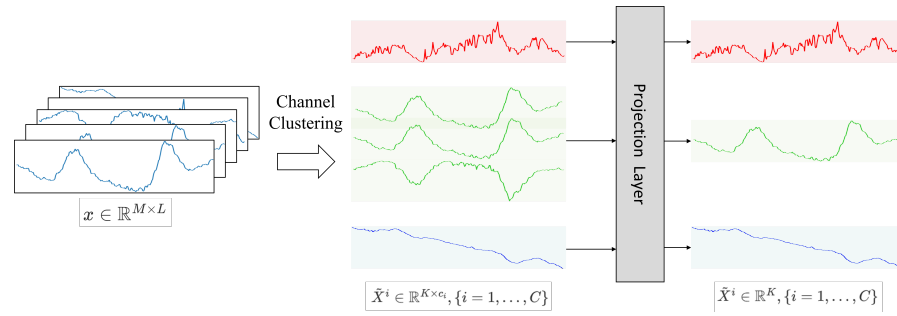


Figure 2. Channels are grouped into C clusters; intra-cluster channels use the CD strategy, while inter-cluster channels are independently passed through the projection layer.

Time-Series Decomposition. The periodicity of time series can vary significantly across different channels. To address this, we apply the fast Fourier transform (FFT) to assess the periodicity of each channel. If the periodicity exceeds a predefined threshold F_{th} , then the data are decomposed into trend, seasonal, and residual components using STL decomposition [28]. These components are treated as separate features and passed into the embedding layer. For channels with weak periodicity, we utilize the original series directly in order to preserve its dynamic characteristics. This differentiated decomposition approach enables the model to better capture complex time series features, improving both the accuracy and robustness of multivariate time series forecasting.

For a strongly periodic time series $X^i \in \mathbb{R}^K$ with i dimensions and a given time step K , the original data can be decomposed into seasonal, trend, and residual components by additive decomposition:

$$X^i = X_T^i + X_S^i + X_R^i$$

where X_S^i represents the periodicity of the time series, X_T^i represents the overall upward or downward trend or state of the time series within a given time step, and X_R^i is the remaining portion after removing the seasonal and trend components.

Patching. Unlike traditional LLMs, which process discrete word tokens, time series data require additional methods for extracting meaningful semantic information from individual time points. Here, we adopt the patching approach proposed by [10]. This approach significantly extends the historical time horizon while minimizing information redundancy and improving model efficiency.

3.2. Modal Alignment

Large language models operate on discrete tokens, while time series data are inherently continuous; thus, aligning time series data with natural language modalities is a crucial step in unlocking the full potential of these models for time series processing. As depicted in Figure 3, we adopt the reprogramming scheme introduced by [25], which transforms input time series into a natural language modality representation. This alignment enhances the backbone network's ability to comprehend temporal patterns.

The scheme uses a multihead cross-attention layer to align the time series patch X_P with the pretrained word embeddings $E \in \mathbb{R}^{V \times D}$ in the LLM backbone network, where V denotes the size of the vocabulary and D is the hidden dimension of the backbone model. To reduce the space required for reprogramming, we maintain a small set of textual prototypes, denoted as $E' \in \mathbb{R}^{V' \times D}$, where $V' \ll V$. The prototypes are obtained using learnable linear projections.

Specifically, for each header $h = \{1, \dots, H\}$, we define the query matrix $Q_h = X_P W_Q^h$, key matrix $K_h = E' W_K^h$, and value matrix $V_h = E' W_V^h$. The operation of reprogramming the time series patch in each attention head is defined as

$$Z_h = \text{Attention}(Q_h, K_h, V_h) = \text{softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_h}}\right) V_h.$$

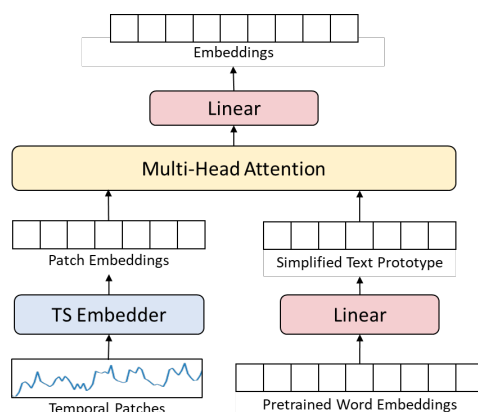


Figure 3. Reprogramming architecture.

3.3. Adaptive Prompts for LLMs

To enhance the ability of large language models (LLMs) to understand prediction tasks and leverage domain knowledge encoded during pretraining [36], we design adaptive prompts that guide the LLM in performing precise time series forecasting. These prompts are tailored to the input series, and consist of the following three key components:

- **Dataset Context:** Provides background information about the input time series, incorporating semantic features along with domain-specific knowledge such as chemical- or industry-specific insights.
- **Data Features:** Includes statistical metrics (e.g., mean and standard deviation), temporal attributes (e.g., periodicity and trends), and correlations derived during preprocessing.
- **Task Instructions:** These instructions explicitly guide the LLM to perform specific predictive tasks by leveraging the provided data features.

An example of such a prompt is shown in Figure 4, which illustrates how the prompt incorporates dataset context, data features, and task instructions for a specific time series forecasting problem.

This dataset encompasses the volumetric percentage (%) of various gaseous components over specific time intervals, which are key reactants or products involved in cracking or reforming processes within petroleum refining.

[Domain]: The dataset contains following features:

- **Hydrogen, Oxygen, Nitrogen:** These gases participate in chemical reactions. Hydrogen, for instance, is used in hydroprocessing to help remove impurities like sulfur from heavier hydrocarbons.
- **Liquefied Petroleum Gas:** These show how much cracking has occurred and the types of products formed.
- **C3 and above:** This indicates the proportion of heavier hydrocarbons produced during refining.

[Instruction]: Forecast the next **<H>** steps given the previous **<K>** steps information.

[Statistics]: The dataset includes a minimum concentration of **<min_val>**, a maximum concentration of **<max_val>**, and a median concentration of **<median_val>**. The overall trend in the concentration levels is **<upward or downward>**.

Figure 4. Prompt example.

The adaptive prompts are further customized based on the periodicity of the data, leveraging the temporal decomposition module described in Section 3.1. For strongly periodic data, decomposition into trend, seasonal, and residual components enables the LLM to process the embedded temporal features directly. In these cases, prompts are streamlined to avoid redundancy. Conversely, weakly periodic data, which are not decomposed, require more detailed prompts to help the LLM identify temporal patterns. This differentiation ensures optimal utilization of the LLM's inference capabilities, enhancing both its temporal pattern recognition and overall prediction performance.

To fully leverage temporal patterns in the data, we concatenate the adaptive prompts with their corresponding components and input them to the pretrained LLM. For highly periodic channel data, the trend and seasonal components are combined to construct the final input representation $x = x_T \oplus x_S \oplus x_R$, where $\oplus$ denotes the concatenation operation. For weakly periodic data, the original time series embedding x is directly input to the LLM without decomposition. This flexible input strategy effectively aligns with the varying characteristics of the data, enhancing the accuracy and robustness of time series forecasting.

4. Results

4.1. Experiment Setup

Dataset. We evaluated seven popular publicly available datasets that have been widely used for benchmarking long-term forecasting models: ETTh1, ETTh2, ETTm1, ETTm2, Weather, Electricity, and Traffic. Additionally, we introduced two real-world refinery datasets: the LIMS dataset, which records the variation in reactant concentrations during the refining process, and the PI dataset, which tracks changes in each generator sensor throughout the same process.

Baseline and Metrics. We selected a series of recent and representative baseline models, including the MLP-based model DLinear [11], the transformer-based models PatchTST [10], Informer [16], and Autoformer [17], and the recent LLM-based generalized temporal prediction models GPT4TS [22], TEMPO [26], and Time-LLM [25]. The evaluation metrics were the mean square error (MSE), mean absolute error (MAE), and relative standard error (RSE).

Experimental Setup. The input time series length T was set to 336 steps, and we considered four different prediction horizons of $H \in \{96, 192, 336, 720\}$. To ensure fairness, all LLM-based methods used GPT-2 [9] as the backbone model. We trained the models using the AdamW optimizer and selected the model with the lowest average validation loss for testing. For fair evaluation, we used the optimal experimental configuration as provided in the official code to implement the base models as well as to adhere to TimesNet's [30] experimental settings. All the experiments were implemented with PyTorch 2.3.1 on a single NVIDIA A100 40GB GPU, and each experiment was repeated at least three times using different random seeds.

4.2. Few-Shot Forecasting

LLMs have demonstrated strong performance in few-shot learning; therefore, we evaluated the few-shot capability of ADTime on both the public and refinery datasets. Following the classic experimental setup [22], we made predictions using 5% of the training data. The few-shot results for our model and the baselines on the public and domain-specific datasets are presented in Tables 1 and 2.

Table 1. Few-shot learning on 5% training data from the public datasets. All results are averaged from four different forecasting horizons: $H \in \{96, 192, 336, 720\}$. Lower values indicate better performance. **Red:** best; **Blue:** second-best.

Methods		Ours			TEMPO			Time-LLM			GPT4TS			PatchTST			Dlinear			Autoformer			Informer		
Metric		MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE
ETTh1	96	0.602	0.514	0.737	0.642	0.550	1.045	0.535	0.672	0.779	0.537	0.525	0.833	0.689	0.574	0.818	0.617	0.584	0.987	0.790	0.650	0.850	1.010	0.740	0.960
	192	0.686	0.580	0.742	0.805	0.598	1.224	0.971	0.678	0.938	0.732	0.615	0.781	0.859	0.667	0.846	0.714	0.621	0.766	0.975	0.736	0.902	1.211	0.840	1.058
	336	0.926	0.660	0.920	1.323	0.803	1.117	1.355	0.817	1.110	1.108	0.760	0.964	1.175	0.793	0.992	0.947	0.762	0.791	0.965	0.729	0.899	1.621	0.895	1.182
	avg	0.738	0.584	0.800	0.923	0.651	1.129	0.954	0.722	0.942	0.792	0.634	0.859	0.907	0.678	0.885	0.759	0.655	0.848	0.910	0.705	0.884	1.281	0.825	1.067
ETTh2	96	0.388	0.412	0.502	0.407	0.429	0.543	0.395	0.415	0.507	0.531	0.494	0.723	0.401	0.422	0.637	0.436	0.487	0.673	0.590	0.590	0.610	3.620	1.520	1.520
	192	0.398	0.409	0.543	0.440	0.443	0.569	0.516	0.476	0.576	0.391	0.414	0.794	0.412	0.446	0.814	0.603	0.498	0.698	0.519	0.420	0.716	3.693	1.314	1.649
	336	0.394	0.411	0.518	0.447	0.454	0.601	0.524	0.485	0.578	0.393	0.393	0.736	0.422	0.451	0.840	0.894	0.514	0.909	0.535	0.431	0.747	3.453	1.660	2.021
	avg	0.393	0.411	0.521	0.431	0.442	0.571	0.478	0.459	0.554	0.439	0.434	0.751	0.412	0.440	0.764	0.645	0.499	0.760	0.548	0.480	0.691	3.588	1.498	1.730
ETTm1	96	0.411	0.410	0.614	0.535	0.478	0.835	0.478	0.438	0.658	0.399	0.415	0.631	0.429	0.429	0.680	0.422	0.474	0.815	0.780	0.610	0.840	1.040	0.760	0.970
	192	0.455	0.450	0.662	0.549	0.495	0.855	0.479	0.451	0.659	0.490	0.466	0.639	0.599	0.533	0.706	0.427	0.434	0.596	0.907	0.677	0.869	1.476	0.725	0.936
	336	0.435	0.431	0.646	0.635	0.531	0.888	0.443	0.424	0.634	0.587	0.519	0.699	0.683	0.592	0.754	0.493	0.472	0.641	0.923	0.695	0.878	1.878	0.839	0.949
	720	0.487	0.467	0.694	0.667	0.553	1.051	0.593	0.512	0.733	0.728	0.591	0.781	1.010	0.711	0.920	0.563	0.525	0.687	1.036	0.762	0.931	1.096	0.844	1.110
	avg	0.447	0.439	0.654	0.597	0.514	0.907	0.498	0.456	0.671	0.551	0.498	0.687	0.680	0.566	0.765	0.476	0.476	0.685	0.912	0.686	0.879	1.372	0.792	0.991
ETTm2	96	0.173	0.253	0.374	0.225	0.307	0.489	0.240	0.311	0.397	0.194	0.277	0.441	0.219	0.299	0.468	0.320	0.380	0.460	0.370	0.340	0.490	3.590	1.510	1.530
	192	0.227	0.248	0.396	0.302	0.350	0.453	0.277	0.333	0.426	0.250	0.288	0.486	0.263	0.322	0.506	0.327	0.381	0.627	0.311	0.322	0.575	3.598	1.475	1.317
	336	0.250	0.325	0.454	0.348	0.382	0.510	0.342	0.375	0.472	0.296	0.346	0.542	0.298	0.341	0.560	0.379	0.424	0.694	0.331	0.333	0.603	3.792	1.123	1.380
	720	0.407	0.420	0.515	0.427	0.421	0.562	0.471	0.445	0.552	0.402	0.416	0.696	0.371	0.393	0.659	0.464	0.488	0.794	0.385	0.376	0.675	3.623	1.503	1.548
	avg	0.264	0.311	0.435	0.325	0.365	0.504	0.332	0.366	0.462	0.286	0.332	0.541	0.288	0.339	0.549	0.373	0.418	0.644	0.349	0.343	0.586	3.651	1.425	1.444
Weather	96	0.183	0.236	0.438	0.193	0.239	0.622	0.193	0.245	0.579	0.185	0.244	0.450	0.192	0.242	0.443	0.280	0.330	0.700	0.300	0.360	0.720	0.510	0.510	0.940
	192	0.193	0.257	0.604	0.246	0.289	0.706	0.244	0.282	0.650	0.243	0.283	0.648	0.264	0.306	0.676	0.227	0.281	0.627	0.381	0.401	0.813	0.708	0.599	1.108
	336	0.284	0.319	0.718	0.326	0.343	0.851	0.290	0.314	0.708	0.301	0.330	0.721	0.314	0.339	0.736	0.279	0.324	0.694	0.404	0.411	0.835	0.680	0.583	1.083
	720	0.347	0.371	0.791	0.372	0.381	0.935	0.378	0.373	0.809	0.371	0.381	0.802	0.395	0.396	0.827	0.364	0.388	0.794	0.431	0.425	0.863	0.633	0.552	1.047
	avg	0.252	0.296	0.638	0.284	0.313	0.778	0.277	0.303	0.686	0.275	0.309	0.655	0.291	0.321	0.671	0.288	0.331	0.704	0.379	0.399	0.808	0.633	0.561	1.044
ECL	96	0.182	0.304	0.408	0.180	0.292	0.405	0.215	0.328	0.461	0.146	0.242	0.382	0.144	0.239	0.379	0.230	0.261	0.392	0.510	0.550	0.710	1.330	0.930	1.150
	192	0.158	0.241	0.382	0.182	0.285	0.447	0.225	0.334	0.471	0.172	0.273	0.413	0.225	0.329	0.472	0.157	0.252	0.394	0.436	0.495	0.656	1.326	0.947	1.145
	336	0.178	0.277	0.414	0.209	0.309	0.524	0.233	0.341	0.481	0.187	0.286	0.430	0.320	0.415	0.563	0.268	0.268	0.414	0.440	0.503	0.660	1.325	0.950	1.146
	720	0.224	0.312	0.449	0.279	0.355	0.574	0.283	0.377	0.531	0.241	0.331	0.490	0.292	0.382	0.539	0.218	0.309	0.465	0.519	0.551	0.719	1.279	0.926	1.128
	avg	0.186	0.284	0.413	0.213	0.310	0.488	0.239	0.345	0.486	0.186	0.283	0.429	0.245	0.341	0.488	0.218	0.272	0.416	0.476	0.525	0.686	1.315	0.938	1.142
Traffic	96	0.392	0.320	0.603	0.438	0.311	0.625	0.653	0.443	0.669	0.434	0.320	0.659	0.413	0.286	0.642	0.460	0.346	0.634	0.875	0.513	1.003	1.730	0.900	1.090
	192	0.419	0.311	0.574	0.496	0.355	0.687	0.690	0.468	0.686	0.4597	0.329	0.560	0.441	0.298	0.548	0.447	0.314	0.572	0.876	0.538	0.772	1.547	0.818	1.027
	336	0.437	0.314	0.599	0.503	0.356	0.681	0.690	0.465	0.683	0.479	0.334	0.569	0.460	0.309	0.557	0.484	0.340	0.572	0.962	0.585	0.806	1.580	0.829	1.033
	avg	0.416	0.315	0.592	0.479	0.341	0.664	0.677	0.459	0.679	0.457	0.328	0.596	0.438	0.298	0.583	0.464	0.333	0.592	0.904	0.545	0.861	1.619	0.849	1.050

For the ETTh and Traffic datasets, the 720 horizon is not included because 5% of the time series was insufficient to constitute a training set.

The results show that ADTime outperforms all baseline models. After testing on seven public time series datasets, ADTime achieved the best performance in 22 out of 25 instances, outperforming the second-best model by 9.5%, 6.1%, and 8.5% in terms of MSE, MAE, and RSE, respectively. In our evaluation on seven domain-specific datasets, ADTime achieved state-of-the-art (SOTA) performance in 19 out of 20 instances, with improvements of 17.1%, 13.5%, and 10.6% over GPT4TS in terms of MSE, MAE, and RSE, respectively.

It is worth noting that PatchTST and DLinear outperform the other baselines on the public datasets, particularly the larger-scale Traffic and Electricity datasets. On the domain-specific datasets, the LLM-based approaches (Time-LLM and GPT4TS) show significantly better performance than the traditional models. This supports our motivation for using LLMs in time series forecasting. More specifically, these results suggest that prompt-based learning can enhance LLMs' understanding of domain knowledge, leading to improved forecasting abilities.

Table 2. Few-shot learning on 5% training data from the refinery datasets. All results are averaged from four different forecasting horizons: $H \in \{96, 192, 336\}$. Lower values indicate better performance. **Red:** best; **Blue:** second-best.

Methods		Ours			TEMPO			Time-LLM			GPT4TS			PatchTST			Dlinear			Autoformer			Informer		
Metric		MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE
PI1	96	1.226	0.986	0.118	1.234	1.130	0.119	1.248	1.060	0.118	1.175	1.035	0.115	1.971	1.435	0.149	1.669	1.353	0.137	1.803	1.712	0.136	5.968	3.777	2.014
	192	0.971	0.992	0.120	1.628	1.277	0.143	1.534	1.215	0.132	1.669	1.311	0.137	1.593	1.331	0.134	2.196	2.817	0.285	1.965	1.733	0.149	5.926	3.749	1.725
	336	1.929	1.334	0.142	2.022	1.423	0.157	1.938	1.404	0.148	2.051	1.447	0.152	3.649	2.466	0.203	1.963	2.524	0.347	2.112	1.799	0.155	5.908	3.743	1.628
	avg	1.375	1.104	0.126	1.628	1.277	0.140	1.573	1.226	0.133	1.632	1.264	0.135	2.405	1.744	0.162	1.943	2.231	0.256	1.960	1.748	0.146	5.934	3.756	1.789
PI2	96	0.410	0.651	0.366	0.442	0.724	0.412	0.417	0.670	0.393	0.458	0.683	0.412	0.430	0.773	0.399	0.382	0.710	0.376	0.489	1.009	0.409	2.730	1.821	1.540
	192	0.552	0.711	0.371	0.555	0.842	0.438	0.509	0.764	0.429	0.652	1.016	0.486	0.537	0.800	0.441	0.542	0.869	0.443	0.518	1.091	0.433	2.806	2.210	1.829
	336	0.498	0.677	0.363	0.587	0.879	0.445	0.518	0.809	0.426	0.686	1.005	0.490	0.666	0.930	0.483	0.608	1.084	0.461	0.502	1.106	0.419	2.899	2.661	1.726
	avg	0.487	0.679	0.366	0.528	0.815	0.432	0.481	0.748	0.416	0.599	0.901	0.463	0.544	0.834	0.441	0.511	0.888	0.427	0.503	1.069	0.420	2.812	2.231	1.698
PI3	96	0.989	0.830	0.107	2.016	1.238	0.113	1.360	1.088	0.094	1.096	0.996	0.084	1.572	1.232	0.101	1.683	1.229	0.104	2.236	1.791	0.134	2.568	1.663	0.240
	192	0.966	0.883	0.114	1.953	1.225	0.117	2.226	1.354	0.119	1.266	1.148	0.090	1.898	1.355	0.110	1.627	1.497	0.322	2.960	1.893	0.137	2.680	1.658	0.231
	336	1.329	1.291	0.106	2.139	1.428	0.173	3.716	1.707	0.153	1.986	1.368	0.112	2.187	1.492	0.117	1.547	1.457	0.312	4.181	2.102	0.162	2.695	1.754	0.223
	avg	1.094	1.001	0.109	2.036	1.297	0.134	2.434	1.383	0.122	1.449	1.170	0.095	1.886	1.360	0.109	1.619	1.394	0.246	3.126	1.929	0.145	2.648	1.691	0.231
PI4	96	0.179	0.184	0.364	0.197	0.219	0.395	0.186	0.210	0.363	0.290	0.281	0.454	0.237	0.255	0.410	0.239	0.222	0.397	0.402	0.366	0.534	1.416	0.690	1.002
	192	0.204	0.211	0.356	0.230	0.277	0.420	0.208	0.226	0.384	0.219	0.284	0.448	0.229	0.245	0.404	0.229	0.245	0.404	0.389	0.409	0.526	1.545	0.741	1.048
	336	0.263	0.272	0.454	0.279	0.322	0.465	0.269	0.266	0.437	0.279	0.279	0.445	0.250	0.254	0.421	0.294	0.290	0.457	0.623	0.581	0.665	1.714	0.871	1.103
	avg	0.215	0.222	0.391	0.235	0.273	0.427	0.221	0.234	0.395	0.263	0.281	0.449	0.239	0.251	0.412	0.254	0.252	0.419	0.471	0.452	0.575	1.558	0.767	1.051
PI5	96	0.651	0.256	0.326	0.654	0.265	0.362	0.695	0.262	0.358	0.670	0.268	0.346	0.808	0.309	0.386	0.753	0.287	0.372	0.912	0.391	0.410	5.856	0.925	1.038
	192	0.525	0.274	0.287	0.775	0.298	0.414	0.788	0.284	0.381	0.751	0.389	0.517	0.825	0.311	0.390	0.900	0.302	0.407	1.031	0.439	0.436	5.957	1.000	1.048
	336	0.663	0.372	0.332	0.878	0.327	0.391	0.963	0.347	0.421	0.865	0.318	0.399	0.825	0.311	0.390	1.268	0.418	0.483	1.368	0.675	0.502	5.744	1.033	1.028
	avg	0.613	0.301	0.315	0.769	0.297	0.389	0.816	0.302	0.387	0.762	0.325	0.421	0.819	0.311	0.388	0.974	0.336	0.421	1.104	0.502	0.449	5.853	0.986	1.038
LIMS1	96	0.955	0.573	0.781	1.277	0.702	1.128	1.726	0.864	1.045	1.020	0.606	0.807	1.093	0.638	0.830	2.078	1.113	1.152	1.118	0.690	0.838	2.014	1.102	1.124
	192	1.190	0.647	0.843	0.987	0.600	1.162	1.617	0.808	0.982	1.192	0.655	0.843	1.711	0.847	1.011	2.159	1.147	1.135	1.192	0.716	0.836	2.292	1.226	1.158
	avg	1.073	0.610	0.812	1.132	0.651	1.145	1.671	0.836	1.013	1.106	0.630	0.825	1.402	0.742	0.920	2.119	1.130	1.144	1.155	0.703	0.837	2.153	1.164	1.141
LIMS2	96	0.430	0.458	0.478	0.498	0.477	0.663	0.624	0.556	0.597	0.546	0.489	0.504	0.523	0.490	0.546	2.157	1.131	1.109	0.681	0.607	0.624	1.941	1.104	1.054
	192	0.438	0.449	0.482	0.506	0.527	0.897	0.686	0.579	0.631	0.545	0.480	0.508	0.523	0.483	0.551	1.737	1.025	1.005	0.574	0.533	0.578	2.225	1.188	1.139
	336	0.433	0.478	0.500	0.560	0.572	0.664	0.654	0.546	0.620	0.545	0.475	0.512	0.570	0.499	0.579	2.200	1.146	1.137	0.577	0.518	0.587	2.170	1.170	1.139
	avg	0.434	0.461	0.486	0.521	0.526	0.741	0.655	0.560	0.616	0.545	0.481	0.508	0.539	0.491	0.559	2.031	1.100	1.083	0.610	0.553	0.597	2.112	1.154	1.111

For the LIMS1 dataset, the 360 horizon is not included because 5% of the time series was insufficient to constitute a training set.

4.3. Zero-Shot Forecasting

This task evaluates the cross-dataset adaptability of our proposed scheme, i.e., its temporal prediction performance on dataset A when the model is trained on dataset B without encountering any samples from dataset A. In this section, we evaluate various cross-dataset scenarios using the ETT, PI, and LIMS datasets.

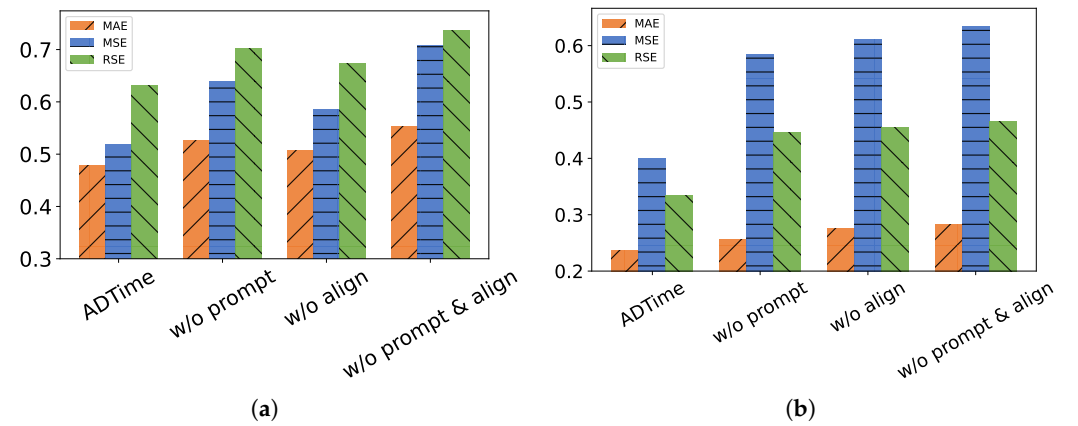
The zero-shot experimental results are shown in Table 3. Our proposed ADTime consistently outperforms all existing baselines across various transfer learning scenarios, demonstrating strong transferability and generalization ability. In particular, the results on real-world refinery datasets PI and LIMS further verify the robustness and scalability of ADTime in practical scenarios. The suboptimal performance of existing models such as Time-LLM and GPT4TS underscores the inherent limitations of generic LLMs in zero-shot settings [37]. These findings suggest that integrating LLMs with well-designed prompts and time series decomposition can significantly improve the accuracy and stability of zero-shot forecasting in real-world domains.

Table 3. Zero-shot learning results. **Red:** best; **Blue:** second-best.

Methods	Ours			Time-LLM			TEMPO			GPT4TS			PatchTST		
Metric	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE	MSE	MAE	RSE
ETTh1→ETTh2	0.367	0.422	0.431	0.375	0.424	0.473	0.392	0.436	0.460	0.406	0.422	0.460	0.380	0.405	0.443
ETTh2→ETTh1	0.611	0.513	0.699	0.640	0.524	0.761	0.742	0.606	0.839	0.703	0.578	0.735	0.665	0.533	0.747
ETTh1→ETTh2	0.203	0.284	0.364	0.217	0.292	0.368	0.238	0.289	0.379	0.264	0.295	0.393	0.296	0.334	0.408
ETTh2→ETTh1	0.442	0.425	0.633	0.478	0.438	0.658	0.515	0.478	0.662	0.769	0.567	0.679	0.568	0.492	0.639
PI5→PI6	0.703	0.248	0.329	0.731	0.264	0.356	1.704	0.260	0.691	0.980	0.340	0.425	0.829	0.326	0.391
PI6→PI5	0.204	0.204	0.364	0.194	0.215	0.371	0.212	0.211	0.352	0.210	0.227	0.386	0.225	0.242	0.400
LIMS1→LIMS2	0.503	0.470	0.504	0.850	0.672	0.696	0.628	0.555	0.670	0.575	0.495	0.522	0.601	0.525	0.592
LIMS2→LIMS1	1.075	0.675	0.810	1.220	0.685	0.877	1.270	0.697	1.116	1.176	0.684	0.794	1.298	0.692	0.872

4.4. Ablation Study

Embedding Variants. In order to better understand the effectiveness of the embedding construction module design in ADTime, we constructed three variants of the model and conducted ablation experiments on the ETTh and PI datasets. The experimental results illustrated in Figure 5 demonstrate that removing either the modality alignment or the prompt module leads to diminished time series forecasting capability on the part of LLMs. Text-aligned ablation indicates that eliminating gaps between the text space and temporal data space leads to better prediction results than simple splicing. The degradation of prediction results in the absence of prompts also validates the significance of prompts in guiding the prediction task.

**Figure 5.** Ablation study of the embedding module: (a) experimental results on the ETTh dataset and (b) experimental results on the PI dataset.

Adaptive Decomposition. We performed an ablation study to verify the significance of adaptive temporal decomposition. Two variants of the module (w/o decomposition and direct decomposition) were constructed and evaluated on the PI and Weather datasets. The experimental results in Figure 6a demonstrate that temporal decomposition is effective for feature extraction. In particular, our adaptive decomposition method selectively identifies significant features, significantly improving the model's performance.

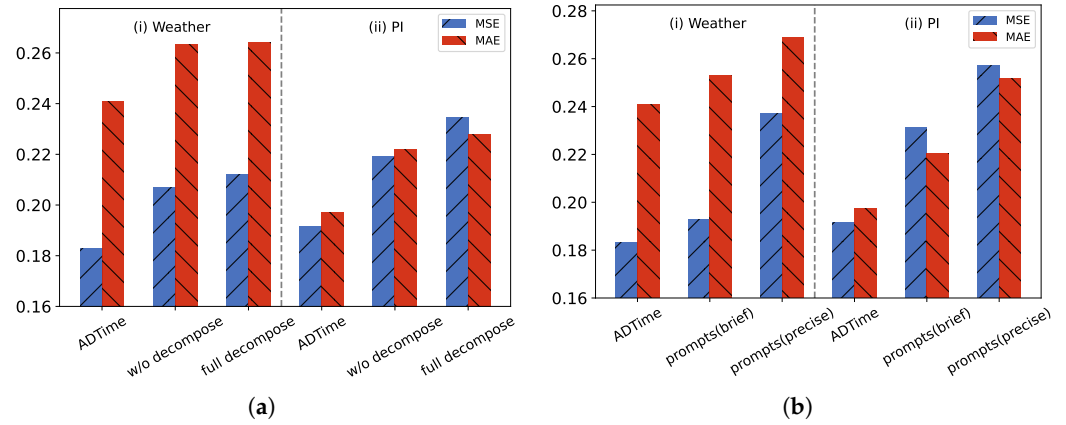


Figure 6. Ablation study of model design: (a) decomposition module ablation experiment on the ETTh and PI datasets and (b) prompt variants on the ETTh and PI datasets.

Furthermore, to validate the generalizability of the adaptive temporal decomposition approach, we modified the PatchTST and TEMPO by incorporating the adaptive module into both. The effectiveness of this modification was evaluated, and results are presented in Table 4. The experimental results confirm that integrating the adaptive temporal decomposition module into existing models consistently enhances their predictive performance. This improvement is evident across different datasets and baseline architectures. It highlights that by selectively focusing on significant components, the module improves the robustness and accuracy of diverse time series models, making it a valuable addition for tackling both general and domain-specific forecasting tasks.

Table 4. Performance comparison of different methods. **Black:** best performance.

Variants	Weather-96		Weather-192		PI5-96		PI5-192	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ADTime	0.183	0.236	0.193	0.247	0.651	0.256	0.525	0.274
TEMPO	0.193	0.239	0.246	0.289	0.654	0.265	0.775	0.298
TEMPO +	0.187	0.237	0.199	0.257	0.652	0.259	0.725	0.286
PatchTST	0.192	0.242	0.264	0.306	0.808	0.309	0.825	0.311
PatchTST +	0.186	0.238	0.213	0.269	0.779	0.277	0.781	0.293

Prompt Variants. We designed two prompt variants in this study: *Prompts(brief)* provides only domain information and task instructions, while *Prompts(precise)* additionally summarizes time series characteristics such as the mean value of the variable and the trend value. The prediction performance of the different prompt designs is illustrated in Figure 6b.

4.5. Model Analysis

LLM Variants. We compared the performance of different large language model variants with varying sizes, as shown in Table 5. The experiment reveals several key findings. First, increasing the model size with the same architecture consistently improves forecasting performance, confirming that larger models are more effective at capturing complex temporal dependencies. Second, the performance of different model architectures varies across datasets, indicating that the knowledge learned by each model may be biased toward specific patterns. These results underscore the importance of selecting a model that aligns with the specific characteristics of the dataset in order to achieve optimal accuracy.

Table 5. Forecasting performance with language model variants; GPT(K) refers to the GPT backbone model with K layers; similarly, LLaMA(K) denotes the LLaMA backbone with K layers. **Black:** best performance.

Model	ETTh1-96		ETTh1-192		ETTh2-96		ETTh2-192		PI4-96		PI4-192		PI5-96		PI5-192	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Llama3.2 (32)	0.561	0.501	0.647	0.538	0.375	0.393	0.372	0.426	0.152	0.166	0.190	0.193	0.679	0.267	0.779	0.284
Llama3.2 (8)	0.592	0.507	0.698	0.551	0.361	0.416	0.377	0.427	0.159	0.178	0.192	0.203	0.694	0.260	0.683	0.280
GPT-2 (12)	0.686	0.554	0.632	0.529	0.323	0.368	0.347	0.409	0.179	0.192	0.200	0.210	0.675	0.257	0.736	0.277
GPT-2 (6)	0.602	0.514	0.686	0.580	0.388	0.412	0.398	0.409	0.179	0.184	0.204	0.211	0.651	0.256	0.525	0.274

Computational Complexity. For the ETTh1-96 dataset with a batch size of 64, ADTime requires approximately 14.25 GB of memory. This is comparable to other LLM-based methods, with a difference of only a few hundred MB. Compared to PatchTST, ADTime’s memory usage is higher by approximately 12.5 GB. However, given a 40 GB GPU, this additional memory cost remains manageable. In terms of processing speed, ADTime achieves 2.68 it/s. While lower than lightweight models such as PatchTST, this is sufficient for real-time processing. Because most time series forecasting tasks operate with hourly or daily sampling intervals, this level of computational efficiency is well within practical requirements, ensuring feasibility in real-world applications. Furthermore, the slight tradeoff in processing speed is justified by the significant improvements in forecasting accuracy and interpretability, making ADTime highly suitable for real-world deployment in scenarios requiring high-stakes decision-making.

T-SNE Visualization. Figure 7 presents the T-SNE visualization of the embedding process for the ETTh dataset at different stages of the multimodal learning pipeline consisting of series and prompt embeddings. Initially, the series embedding shows a sparse distribution of data points, indicating a lack of structured correlation in the time series data. In contrast, prompt embedding exhibits a more compact distribution with the data points clustered together, reflecting the strong correlations established through the prompt-based approach. After the alignment process, the embeddings from both the time series and the prompt overlap closely, demonstrating that text alignment successfully bridges the gap between the temporal and textual spaces. This result confirms the effective integration of time series and prompt embeddings, enhancing the model’s ability to leverage both modalities in a unified manner.

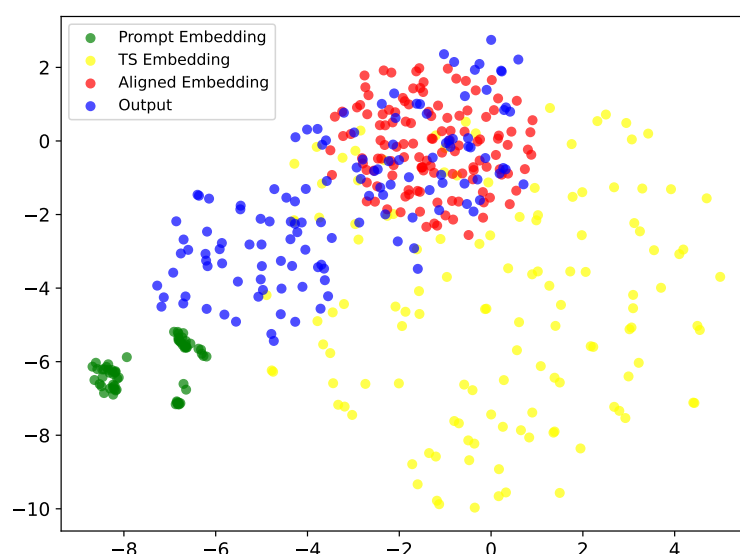


Figure 7. T-SNE visualization of embeddings at different stages for the ETTh dataset.

5. Conclusions

In this paper, we propose ADTime, an adaptive LLM-based framework for multivariate time series forecasting. ADTime leverages an adaptive time series process model to dynamically captures temporal features, followed by a prompt and textual alignment module that transforms structured data into textual representation. By integrating these mechanisms, ADTime achieves state-of-the-art predictive performance across multiple domain-specific datasets, particularly in complex real-world scenarios. Our results demonstrate the effectiveness of adaptive decomposition and clustering in enhancing model robustness and generalization. However, challenges remain in domain adaptation. Future research could explore domain-adaptive fine-tuning to improve robustness. Additionally, leveraging LLMs for end-to-end time series forecasting and reasoning represents a promising direction that could offer deeper insights into temporal dynamics.

Author Contributions: Conceptualization, J.P., Y.Z. and T.L.; methodology, J.P. and T.L.; software, validation, writing—original draft preparation, visualization, J.P.; formal analysis, investigation, resources, K.Q. and Y.Z.; data curation, J.P., T.L. and J.Y.; writing—review and editing, T.L. and J.Y.; supervision, project administration, Q.W. and K.Q.; funding acquisition, K.Q. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Energy R&D Center of Petroleum Refining Technology (RIPP, SINOPEC) under Grant 36800000-23-ZC0607-0107.

Data Availability Statement: One part of the data used in this study is confidential and cannot be shared due to restrictions. The public part of the dataset is available in Google Drive (https://drive.google.com/drive/folders/1ZOYpTUa82_jCcxldTmyr0LXQfvaM9vly), accessed on 10 April 2025.

Acknowledgments: This work was partially supported by the National Energy R&D Center of Petroleum Refining Technology (RIPP, SINOPEC). The authors would like to express their gratitude for the valuable support and resources provided during this research.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Angryk, R.A.; Martens, P.C.; Aydin, B.; Kempton, D.; Mahajan, S.S.; Basodi, S.; Ahmadzadeh, A.; Cai, X.; Filali Boubrahimi, S.; Hamdi, S.M.; et al. Multivariate time series dataset for space weather data analytics. *Sci. Data* **2020**, *7*, 227. [CrossRef] [PubMed]
2. Wang, Y.; Huang, N.; Li, T.; Yan, Y.; Zhang, X. Medformer: A Multi-Granularity Patching Transformer for Medical Time-Series Classification. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 36314–36341.
3. Li, L.; Su, X.; Zhang, Y.; Lin, Y.; Li, Z. Trend modeling for traffic time series analysis: An integrated study. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 3430–3439. [CrossRef]
4. Zhu, Z.; Chen, W.; Xia, R.; Zhou, T.; Niu, P.; Peng, B.; Wang, W.; Liu, H.; Ma, Z.; Gu, X.; et al. Energy forecasting with robust, flexible, and explainable machine learning algorithms. *AI Mag.* **2023**, *44*, 377–393. [CrossRef]
5. Yu, X.; Chen, Z.; Ling, Y.; Dong, S.; Liu, Z.; Lu, Y. Temporal Data Meets LLM—Explainable Financial Time Series Forecasting. *arXiv* **2023**, arXiv:2306.11025.
6. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
7. Liu, T.; Low, B.K.H. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks. *arXiv* **2023**, arXiv:2305.14201.
8. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. Llama: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971.
9. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
10. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
11. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are transformers effective for time series forecasting? *AAAI Conf. Artif. Intell.* **2023**, *37*, 11121–11128. [CrossRef]

12. Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; Sun, L. Transformers in time series: A survey. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, Macao, China, 19–25 August 2023; pp. 6778–6786.
13. Liu, Y.; Wu, H.; Wang, J.; Long, M. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 9881–9893.
14. Kitaev, N.; Kaiser, Ł.; Levskaya, A. Reformer: The efficient transformer. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 30 April 2020.
15. Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; Long, M. itransformer: Inverted transformers are effective for time series forecasting. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2023.
16. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. *AAAI Conf. Artif. Intell.* **2021**, *35*, 11106–11115. [[CrossRef](#)]
17. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22419–22430.
18. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In Proceedings of the International conference on machine learning. PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 27268–27286.
19. Chen, P.; Zhang, Y.; Cheng, Y.; Shu, Y.; Wang, Y.; Wen, Q.; Yang, B.; Guo, C. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. *arXiv* **2024**, arXiv:2402.05956.
20. Jin, M.; Zhang, Y.; Chen, W.; Zhang, K.; Liang, Y.; Yang, B.; Wang, J.; Pan, S.; Wen, Q. Position: What Can Large Language Models Tell Us about Time Series Analysis. In Proceedings of the Forty-first International Conference on Machine Learning, Vienna, Austria, 21–27 July 2024.
21. Liu, C.; Yang, S.; Xu, Q.; Li, Z.; Long, C.; Li, Z.; Zhao, R. Spatial-temporal large language model for traffic prediction. In Proceedings of the 2024 25th IEEE International Conference on Mobile Data Management (MDM), Brussels, Belgium, 24–27 June 2024; pp. 31–40.
22. Zhou, T.; Niu, P.; Sun, L.; Jin, R. One fits all: Power general time series analysis by pretrained lm. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 43322–43355.
23. Liu, C.; Xu, Q.; Miao, H.; Yang, S.; Zhang, L.; Long, C.; Li, Z.; Zhao, R. TimeCMA: Towards LLM-Empowered Time Series Forecasting via Cross-Modality Alignment. In Proceedings of the AAAI, Dubai, United Arab Emirates, 20–22 May 2025.
24. Sun, C.; Li, H.; Li, Y.; Hong, S. TEST: Text prototype aligned embedding to activate LLM’s ability for time series. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
25. Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J.Y.; Shi, X.; Chen, P.Y.; Liang, Y.; Li, Y.F.; Pan, S.; et al. Time-llm: Time series forecasting by reprogramming large language models. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
26. Cao, D.; Jia, F.; Arik, S.O.; Pfister, T.; Zheng, Y.; Ye, W.; Liu, Y. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
27. Wang, S.; Wu, H.; Shi, X.; Hu, T.; Luo, H.; Ma, L.; Zhang, J.Y.; Zhou, J. Timemixer: Decomposable multiscale mixing for time series forecasting. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
28. Cleveland, R.B.; Cleveland, W.S.; McRae, J.E.; Terpenning, I. STL: A seasonal-trend decomposition. *J. Off. Stat.* **1990**, *6*, 3–73.
29. Oreshkin, B.N.; Carpov, D.; Chapados, N.; Bengio, Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
30. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
31. Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A.X.; Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022.
32. Zhang, Y.; Yan, J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
33. Han, L.; Ye, H.J.; Zhan, D.C. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 7129–7142. [[CrossRef](#)]
34. Wang, X.; Zhou, T.; Wen, Q.; Gao, J.; Ding, B.; Jin, R. CARD: Channel aligned robust blend transformer for time series forecasting. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
35. Chen, J.; Lenssen, J.E.; Feng, A.; Hu, W.; Fey, M.; Tassioulas, L.; Leskovec, J.; Ying, R. From Similarity to Superiority: Channel Clustering for Time Series Forecasting. *arXiv* **2024**, arXiv:2404.01340.

36. Li, X.L.; Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing; Volume 1: Long Papers; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 4582–4597.
37. Gruver, N.; Finzi, M.; Qiu, S.; Wilson, A.G. Large language models are zero-shot time series forecasters. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 861.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.