# TimeCMA: Towards LLM-Empowered Multivariate Time Series Forecasting via Cross-Modality Alignment

**Chenxi Liu[1], Qianxiong Xu[1*], Hao Miao[2], Sun Yang[3], Lingzheng Zhang[4], Cheng Long[1], Ziyue Li[5*], Rui Zhao [6]**

[1]S-Lab, Nanyang Technological University, Singapore
[2]Aalborg University, Denmark
[3]Peking University, China
[4]HKUST (Guangzhou), China
[5]University of Cologne, Germany
[6]SenseTime Research, China
{chenxi.liu, qianxiong.xu, c.long}@ntu.edu.sg, haom@cs.aau.dk, 2201210484@stu.pku.edu.cn, lingzhengzhang01@gmail.com, zlibn@wiso.uni-koeln.de, zhaorui@sensetime.com

## Abstract

Multivariate time series forecasting (MTSF) aims to learn temporal dynamics among variables to forecast future time series. Existing statistical and deep learning-based methods suffer from limited learnable parameters and small-scale training data. Recently, large language models (LLMs) combining time series with textual prompts have achieved promising performance in MTSF. However, we discovered that current LLM-based solutions fall short in learning *disentangled* embeddings. We introduce TimeCMA, an intuitive yet effective framework for MTSF via cross-modality alignment. Specifically, we present a dual-modality encoding with two branches: the time series encoding branch extracts *disentangled yet weak* time series embeddings, and the LLM-empowered encoding branch wraps the same time series with text as prompts to obtain *entangled yet robust* prompt embeddings. As a result, such a cross-modality alignment retrieves *both disentangled and robust* time series embeddings, "the best of two worlds", from the prompt embeddings based on time series and prompt modality similarities. As another key design, to reduce the computational costs from time series with their length textual prompts, we design an effective prompt to encourage the most essential temporal information to be encapsulated in the last token: only the last token is passed to downstream prediction. We further store the last token embeddings to accelerate inference speed. Extensive experiments on eight real datasets demonstrate that TimeCMA outperforms state-of-the-arts.

**Code** — https://github.com/ChenxiLiu-HNU/TimeCMA

## Introduction

With the proliferation of scalable mobile sensing, large amounts of time series data, collected across domains such as traffic (Xiao et al. 2022; Miao et al. 2024) and environment (Liu et al. 2024a, 2022), have driven applications such as multivariate time series forecasting (MTSF). MTSF aims to mine temporal dynamics among variables from historical data to predict future time series, enabling users to make proactive decisions, e.g., investment choice (Niu et al. 2020) or weather preparation (Liu et al. 2023).

MTSF methods can be divided into statistical methods (Smith and Demetsky 1997) and deep learning-based methods (Wu et al. 2023; Miao et al. 2022). However, the limited number of learnable parameters and the small-scale training data prevent these methods from achieving better performance and being more robust (Jin et al. 2024b; Liu et al. 2021c,b). Recent advances (Zhou et al. 2023) have incorporated pre-trained LLMs (Radford et al. 2019, 2018) into time series to benefit from the *robust* embeddings learned from abundant language data (Liang et al. 2024).

Existing LLM-based methods for time series forecasting can be categorized by input data types. (1) *Time series-based LLMs* (Zhou et al. 2023; Liu et al. 2024b) replace the LLM's tokenizer with a randomly initialized embedding layer to process time series data. However, this embedding layer initialized with random weights often results in weak embeddings due to a domain gap between time series and language data. (2) *Prompt-based LLMs* (Jin et al. 2024b) introduce prompts with text as additional input to help the LLMs understand the time series forecasting task. The prompts are contextualized within time series with text descriptions to facilitate the data-to-text transformation (Jin et al. 2024a; Xue and Salim 2023) or directly summarize the time series information using pure text (Liu et al. 2024c; Jia et al. 2024; Huang et al. 2024). These prompts are then processed by pre-trained LLMs to obtain robust embeddings, allowing the prompt-based LLMs to outperform existing methods, as evidenced in Table 1.

Although prompt-based LLMs have achieved notable performance, they were challenged by the ***data entanglement issue*** (Chang et al. 2024). Specifically, existing methods (Liu et al. 2024c; Jia et al. 2024) concatenate *disentangled yet weak* time series embeddings (Fig. 1(a), upper) with text prompt embeddings (Fig. 1(a), lower). As one stream of existing methods shown in Fig. 1(b), these fused embeddings are then fed into subsequent time series processing stages. However, the output embeddings are entangled, which de-

*Corresponding author

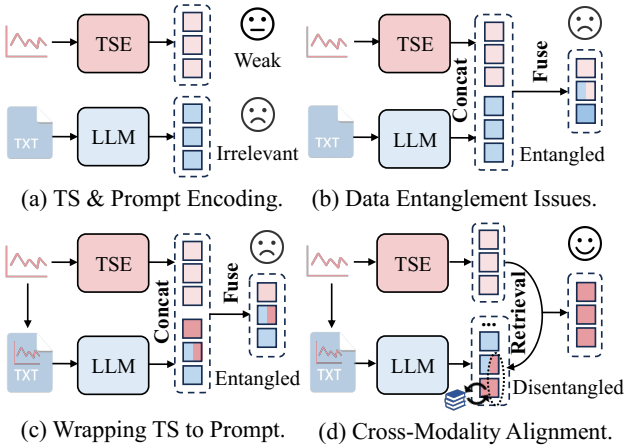



Figure 1: (a) Limits of single-modality models: time series encoder (TSE) offers disentangled yet weak embeddings (in light red); text-only model learns textual embeddings (in blue), irrelevant to time series. (b) Existing models directly fuse two modal embeddings, leading to data-entangled issues. (c) Some tried to wrap time series into prompt, enhancing temporal component in prompt embedding, yet still yielding entanglement. (d) Our method obtains disentangled and robust time series embedding (dark red) via similarity-based retrieval, with last token embeddings stored for efficient forecasting.

grades the forecasting performance because the textual information acts as noise. How to potentially mitigate the noisy textual embedding? As in Fig. 1(c), one attempt is to wrap time series values within text prompts, strengthening the time series embeddings while retaining text to enable LLMs to better understand the time series information as natural language (Xue, Voutharoja, and Salim 2022). Nevertheless, due to the nature of the concatenation method and Transformer blocks within the LLMs, the prompt embeddings become *entangled yet robust*, leading to sub-optimal performance, as in Fig. 3. To address this challenge, we propose that only the ***disentangled and robust time series embeddings*** from LLMs are optimal for MTSF, and this can be easily achieved by our intuitive cross-modality alignment design via similarity-based retrieval for enhancing forecasting, as in Fig. 1(d).

Overall, we present an LLM-empowered framework for multivariate time series forecasting via cross-modality alignment, called **TimeCMA**. It has a dual-modality encoding module with a time series encoding branch and an LLM-empowered encoding branch, a cross-modality alignment module, and a time series forecasting module. The time series encoding branch extracts variable embeddings from historical time series data. The LLM-empowered prompt encoding branch wraps the same time series as prompts to obtain embeddings with well-trained knowledge. Then, the cross-modality alignment module is designed to integrate the two groups of embeddings. Intuitively, as in Fig. 5, the time series embeddings (light red) would naturally have stronger correlations with the time series component (dark red) in the prompt embeddings (mixed color). Therefore, the robust time series components are retrieved from the prompt embeddings based on channel-wise similarity and then aggregated into the original ones (light red) to enhance forecasting.

Nonetheless, prompt-based LLMs suffer from high computational costs and slow inference speeds because: *(i) The characteristics of multivariate time series (MTS) data*: unlike 1D prompt data (with $N$ tokens), MTS data has two dimensions: variable and time (with $N$ variables and $T$ timestamps), causing a substantial computational load. *(ii) High computational burden of LLM outputs.* Despite attempts to reduce computational costs by freezing partial or all of the LLM's parameters, prompt-based LLMs remain computationally expensive because multi-head attention in LLMs generate high-dimensional outputs and require substantial computational power. *(iii) Repetitive processings with the frozen LLMs*: during training, existing prompt-based LLM (Jin et al. 2024a) performs online processing with the frozen LLMs. Consequently, each training sample is processed repetitively by the LLM in each training epoch, though the obtained embeddings remain unchanged due to the frozen parameters. Therefore, the inference speed is considerably slower.

To ease the computational burden, we further propose the last token embedding storage. **(1) The last token is enough**: in the prompt, we independently wrap time series data of each variable to preserve the characteristics of MTS data; then, we tailor the prompt design so that the LLM is instructed to encapsulate vital temporal essences into the last token of each prompt. By only feeding this embedding to align with the time series, we can reduce the computational cost. **(2) Offline storage**: we store the last token embedding to avoid repetitive processing with frozen LLM, thereby accelerating the inference speed. Our contributions are:

- We identify data entanglement issues in the embeddings of dual-modality LLMs for time series forecasting and proposed a TimeCMA framework to learn disentangled embeddings from LLM with text-time series data.
- The cross-modality alignment module retrieves disentangled and robust time series embeddings from the LLM-empowered prompt embeddings via channel-wise similarity to enhance forecasting.
- We tailor the last token of each prompt to reduce the computational costs. We then store these last token embeddings to avoid repetitive processings with the frozen LLM for efficient forecasting.
- Extensive experiments on eight datasets demonstrate that TimeCMA outperforms state-of-the-arts.

## Related Work

Deep learning models have shown crucial promise in time series forecasting. Convolutional neural networks (CNNs) simultaneously capture variable and temporal correlations (Wu et al. 2023; Jin et al. 2022), while Transformers excel due to their powerful learning capabilities. Early Transformer-based methods (Zhang et al. 2021; Zhou et al. 2022) treat multiple variables at the same timestamp as a single temporal token, which often leads to suboptimal performance by conflating unrelated variables. PatchTST (Nie et al. 2023) mitigates this issue with a channel-independent configuration but overlooks inter-variable dependencies, resulting in longer training times and weaker performance on datasets with many variables.

iTransformer (Liu et al. 2023) addresses these limitations by treating independent time series as tokens to better capture multivariate correlations. Despite these advances, existing deep learning methods remain constrained by limited parameterization and small-scale training data (Cai et al. 2024; Liu et al. 2024d, 2021a; Chen, Wang, and Liu 2020).

Recently, large language models (LLMs) have achieved superior performance in time series analysis, benefiting from extensive parameterization and large-scale training data (Gruver et al. 2023; Jin et al. 2024b; Yang et al. 2024). LLM-based forecasting methods can be categorized as time series-based or prompt-based, depending on whether prompts are included in the input. Time series-based LLMs fine-tune models for univariate (Zhou et al. 2023) or multivariate forecasting (Liu et al. 2024b) by replacing the tokenizer with a randomly initialized embedding layer. However, embeddings trained on limited data often suffer due to the domain gap between time series and language data. To address this, prompt-based LLMs incorporate prompts as full or partial input. Early works (Xue, Voutharoja, and Salim 2022; Xue and Salim 2023) explored pure prompting techniques for time series forecasting. Subsequent studies demonstrated that combining time series with prompts (Jin et al. 2024a; Liu et al. 2024c; Cao et al. 2024) or leveraging pre-trained LLM knowledge (Pan et al. 2024; Sun et al. 2024) can enhance performance. However, these approaches still face challenges such as data entanglement and high computational costs.

# Preliminaries

**Multivariate Time Series**. It is denoted as $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\} \in \mathbb{R}^{L \times N}$, where $L$ is the number of time steps and $N$ is the number of variables.

**Prompt**. We wrap the time series $\mathbf{X} \in \mathbb{R}^{L \times N}$ into prompts $\mathbf{P}_S = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \in \mathbb{R}^{S \times N}$ along with variables, as depicted in Fig. 2. Each prompt $\mathbf{p}_i$ has $S$ elements containing words and time series values. In the prompt, the $<$ *italic* $>$ elements represent time information, such as timestamps and frequency. The $<$ *color* $>$ elements denote time series values of $L$ timesteps. The last value that summarizes temporal information is quantified by the total trend $\Delta_T$, defined as:

$$\Delta_T = \sum_{i=1}^{T-1} \delta v_i, \tag{1}$$

where $\delta v_i = v_{i+1} - v_i$ represents the incremental change between consecutive timesteps.

**Problem Definition.** Given an observation in a multivariate time series $\mathbf{x}_t \in \mathbb{R}^N$, where $t$ is a time step. Our goal is to learn a function using historical data $\mathbf{X}_T = \{\mathbf{x}_{t-T+1:t}\} \in \mathbb{R}^{T \times N}$ with $\mathbf{P}_S$ to forecast future multivariate time series $\widehat{\mathbf{X}}_M = \{\widehat{\mathbf{x}}_{t+1:t+M}\} \in \mathbb{R}^{M \times N}$ over $M$ timesteps.

# Methodology

## Framework Overview

TimeCMA contains three key modules: dual-modality encoding, cross-modality alignment, and time series forecasting, as shown in Fig. 2.

**Dual-Modality Encoding** include a time series encoding branch and an LLM-empowered encoding branch, to effectively learn embeddings for input time series and prompts.

*Time Series Encoding Branch* consists of an inverted embedding layer and a time series encoder. The inverted embedding treats an entire variable's time series as a token (Liu et al. 2024b), generating token embeddings that are fed into a Pre-LN Transformer encoder (Xiong et al. 2020).

*LLM-Empowered Encoding Branch* comprises a frozen LLM, and a prompt encoder with the same architecture as that in the time series encoder. The frozen LLM extracts prompt embeddings with sufficient information extracted from the times series, while the prompt encoder refines these embeddings across multiple variables.

**Cross-Modality Alignment** aggregates the dual modalities. The purpose is to retrieve time series embeddings from the prompt embeddings based on their similarity.

**Time Series Forecasting** has a multivariate Transformer decoder similar to that in the lightweight Pre-LN Transformer, which decodes the aligned time series embeddings and then inputs them into a projection function for future forecasting.

## Dual-Modality Encoding

**Time Series Encoding Branch** The time series branch employs an inverted embedding (Liu et al. 2024b), which defines the entire time series of a variable as a token, to generate token embeddings. The time series encoder effectively captures complex temporal dependencies between these tokens.

**Inverted Embedding.** Given the time series data $\mathbf{X}_T \in \mathbb{R}^{T \times N}$, the inverted embedding aims to convert $\mathbf{X}_T$ into learnable matrices $\mathbf{H}_T \in \mathbb{R}^{C \times N}$ to capture the temporal dependencies of variables (Liu et al. 2023). The $\mathbf{X}_T$ is initially normalized to have zero mean and unit standard deviation via reversible instance normalization to mitigate the time series distribution shift (Kim et al. 2022). Then, the normalized $\mathbf{X}_T$ is transformed to variable embedding:

$$\mathbf{H}_T = \mathbf{W}_e \mathbf{X}_T + \mathbf{b}_e, \tag{2}$$

where $C$ indicates the hidden dimension of the embedded time series. $\mathbf{W}_e$ and $\mathbf{b}_e$ are the learnable parameters.

**Time Series Encoder.** The variable embeddings $\mathbf{H}_T$ are fed into a lightweight encoder $TSEncoder(\cdot)$. Inspired by the Transformer structure in existing LLMs (Xu et al. 2024), we apply layer normalization first in the encoder, meaning it occurs before both the multi-head attention and feed-forward layers. Compared with the original Transformer, this Pre-LN Transformer has the advantages of being more stable and converging faster (Huang et al. 2023). In $TSEncoder(\cdot)$, the embeddings $\mathbf{H}_T^i$ undergo $i_{\text{th}}$ layer normalization $LN(\cdot)$:

$$\widetilde{\mathbf{H}}_T^i = LN(\mathbf{H}_T^i), \tag{3}$$

$$LN(\mathbf{H}_T^i) = \gamma \odot \frac{\mathbf{H}_T^i - \mu}{\sigma} + \beta, \tag{4}$$

where $\widetilde{\mathbf{H}}_T^i$ represents the intermediate embedding after the $i_{\text{th}}$ layer normalization. $\gamma$ and $\beta$ are learnable scaling and translation parameters. $\mu$ and $\sigma$ represent the mean and standard deviation. $\odot$ denotes element-wise multiplication.

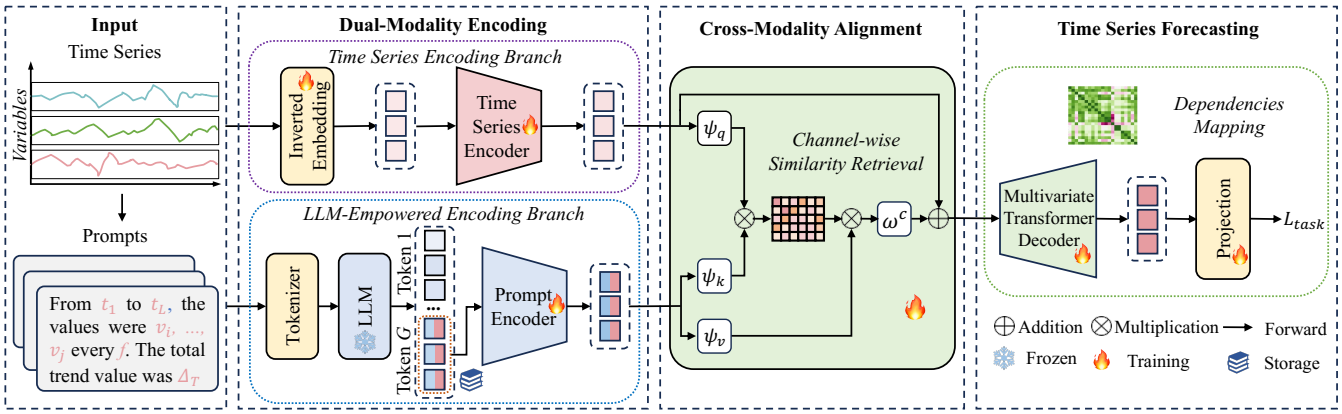



Figure 2: Overall Framework of TimeCMA.

Then, they are processed by the multi-head self-attention mechanism, denoted as $MHSA(\cdot)$. The output, $\overline{\mathbf{H}}_T^i$, is combined with $\mathbf{H}_T^i$ through a residual connection:

$$\overline{\mathbf{H}}_T^i = MHSA(\widetilde{\mathbf{H}}_T^i) + \mathbf{H}_T^i, \tag{5}$$

$$MHSA(\mathbf{H}_T^i) = \rho_o(Attention(\rho_q\mathbf{H}_T^i, \rho_k\mathbf{H}_T^i, \rho_v\mathbf{H}_T^i)), \tag{6}$$

where $\overline{\mathbf{H}}_T^i$ is output of the $i_{\text{th}}$ layer after the $MHSA(\cdot)$. $\rho_o$, $\rho_q$, $\rho_k$, and $\rho_v$ are the linear projections.

Followed by another $LN(\cdot)$. The normalized $\grave{\mathbf{H}}_T^{i+1}$ are then passed through a feed-forward network $FFN(\cdot)$ of fully connected layers that further process the embeddings, then combined with the $\overline{\mathbf{H}}_T^i$ through another residual connection:

$$\grave{\mathbf{H}}_T^{i+1} = LN(\overline{\mathbf{H}}_T^i), \tag{7}$$

$$\overline{\mathbf{H}}_T^{i+1} = FFN(LN(\grave{\mathbf{H}}_T^i)) + \overline{\mathbf{H}}_T^i, \tag{8}$$

where $\grave{\mathbf{H}}_T^i$ represents the intermediate embedding of the $i_{\text{th}}$ layer after the second $LN(\cdot)$. To simplify, $\overline{\mathbf{H}}_T \in \mathbb{R}^{C\times N}$ symbolizes the output of $TSEncoder(\cdot)$.

**LLM-Empowered Encoding Branch** Pre-trained LLMs learn from input tokens, making them more sample-efficient than encoder-only models given the same training data (BehnamGhader et al. 2024). We selected GPT-2 as the LLM to generate the prompt embeddings, which enhance the time series embeddings. The GPT-2 comprises a tokenizer and a GPT-2 model. All parameters in the GPT-2 are frozen.

**Pre-trained LLM.** The tokenizer is responsible for converting prompt input $\mathbf{P}_S \in \mathbb{R}^{S\times N}$ into a series of token IDs $\mathbf{P}_G \in \mathbb{R}^{G\times N}$, where $G$ represents the token ID number in a prompt. Subsequently, these prompt tokens are fed into the GPT-2 model to generate prompt embeddings:

$$\overline{\mathcal{P}}_G^i = MMSA(LN(\mathbf{P}_G^i)) + \mathbf{P}_G^i, \tag{9}$$

$$\mathcal{P}_G^{i+1} = FFN(LN(\overline{\mathcal{P}}_G^i)) + \overline{\mathcal{P}}_G^i, \tag{10}$$

$$MMSA(\mathbf{P}_G^i) = \phi_o(Attention(\phi_q\mathbf{P}_G^i, \phi_k\mathbf{P}_G^i, \phi_v\mathbf{P}_G^i)), \tag{11}$$

where $\overline{\mathcal{P}}_G^i \in \mathbb{R}^{G\times N\times E}$ represents the intermediate representation of the $i_{\text{th}}$ layer after applying the $MMSA(\cdot)$ and the $LN(\cdot)$, $E$ denotes the hidden dimension of the GPT-2. $\mathbf{P}_G^0 = [\mathbf{P}_G + \mathbf{PE}]$, $\mathbf{PE}$ represents the learnable positional encoding. $\phi_o$, $\phi_q$, $\phi_k$, and $\phi_v$ are the linear projections. $\mathcal{P}_G^{i+1} \in \mathbb{R}^{G\times N\times E}$ symbolizes the output of GPT-2 .

**Last Token Embedding Storage**. It is verified that not all tokens are equally important for language model training (Lin et al. 2024; BehnamGhader et al. 2024). The last token in a prompt holds the most comprehensive knowledge due to the masked multi-self attention within the LLMs. Specifically, the representation of the last token at position $G$ is influenced exclusively by the representations of its previous tokens at positions $\{1, 2, \cdots, G-1\}$. Thus, we tailor and store the well-trained last token embeddings $\mathbf{L}_N = \{\mathbf{l}_1, \ldots, \mathbf{l}_N\} \in \mathbb{R}^{N\times E}$ from the $\mathcal{P}_G^{i+1}$ to reduce computational costs.

**Prompt Encoder.** We define prompt encoder as $PromptEncoder(\cdot)$. Its structure follows the decoder in Pre-LN Transformer, identical to $TSEncoder(\cdot)$. We denote the output of $PromptEncoder(\cdot)$ as $\overline{\mathbf{L}}_N \in \mathbb{R}^{N\times E}$.

## Cross-Modality Alignment

To aggregate the time series and the prompt modalities, we design a cross-modality alignment based on channel-wise similarity retrieval. It aims at using *disentangled yet weak* time series embeddings $\overline{\mathbf{H}}_T \in \mathbb{R}^{C\times N}$ to retrieve *disentangled and robust* time series embeddings $\overline{\mathbf{H}}_C \in \mathbb{R}^{N\times E}$ from *entangled and robust* prompt embeddings $\overline{\mathbf{L}}_N \in \mathbb{R}^{C\times N}$.

First, we employ three linear layers $\psi_q, \psi_v, \psi_k$ to transform $\overline{\mathbf{H}}_T$ and $\overline{\mathbf{L}}_N$ to three compact embeddings: $\psi_q(\overline{\mathbf{H}}_T)$, $\psi_v(\overline{\mathbf{L}}_N)$, and $\psi_k(\overline{\mathbf{L}}_N)$. Then, we compute the channel-wise similarity matrix $\mathbf{M}_T \in \mathbb{R}^{C\times E}$ by matrix multiplication followed by softmax:

$$\mathbf{M}_T = F_{\text{softmax}}\left(\psi_q\left(\overline{\mathbf{H}}_T\right) \otimes \psi_k\left(\overline{\mathbf{L}}_N\right)\right), \tag{12}$$

where $\otimes$ denotes matrix multiplication.

We perform channel-wise feature aggregation by restoring the channel dimension through the matrix multiplication of $\psi_v(\overline{\mathbf{L}}_N)$ with $\mathbf{M}_T$. Finally, we get the output by adding $\overline{\mathbf{H}}_T$

to it by matrix addition:

$$\overline{\mathbf{H}}_C = \omega^c \left(\psi_v \left(\overline{\mathbf{L}}_N\right) \otimes \mathbf{M}_T\right) \oplus \overline{\mathbf{H}}_T, \tag{13}$$

where $\omega^c$ is the linear layer and $\oplus$ denotes addition.

Through cross-modality alignment, we transfer the knowledge learned from the pre-trained LLM into time series embeddings, which thus improves the model performance.

## Time Series Forecasting

We design a time series forecasting module including a multivariate Transformer decoder and a projection function. In particular, we input the aligned time series embeddings $\overline{\mathbf{H}}_C$ into the multivariate Transformer decoder $MTDecoder(\cdot)$ to map the dependencies among variables. Finally, we use a projection function for final forecasting.

We first feed the $\overline{\mathbf{H}}_C$ into a layer normalization layer $LN(\cdot)$ to obtain normalized embeddings $\widetilde{\mathbf{H}}^i_C$. Then, we employ a masked multi-self attention layer $MMSA(\cdot)$ with residual connection to obtain $\overline{\mathbf{H}}^i_C$.

Then, $\overline{\mathbf{H}}^i_C$ is fed to the second layer normalization $LN(\cdot)$ followed by a multi-head cross-attention layer $MHCA(\cdot)$:

$$\check{\mathbf{H}}^i_C = MHCA(LN(\overline{\mathbf{H}}^i_C)) + \overline{\mathbf{H}}^i_C, \tag{14}$$

$$MHCA(\mathbf{H}^i_C) = \varsigma_o(Attention(\varsigma_q\overline{\mathbf{H}}^i_C, \varsigma_k\overline{\mathbf{H}}^i_C, \varsigma_v\overline{\mathbf{H}}^i_C)), \tag{15}$$

where $\varsigma_o$, $\varsigma_q$, $\varsigma_k$, and $\varsigma_v$ are linear projections. We apply residual connection to obtain the output $\check{\mathbf{H}}_C$ of $MTDecoder(\cdot)$.

Finally, the $\check{\mathbf{H}}_C$ is input into a projection function for future prediction, which is formulated as follows:

$$\widehat{\mathbf{X}}_M = \mathbf{W}_p\check{\mathbf{H}}_C + \mathbf{b}_p, \tag{16}$$

where $\widehat{\mathbf{X}}_M \in \mathbb{R}^{M\times N}$ denotes the projected embeddings. Finally, we denormalize the $\widehat{\mathbf{X}}_M$.

## Overall Objective Function

The loss function of TimeCMA contains two parts: a prediction loss $L_{pre}$ and a regularization loss $L_{reg}$. We combine them and the overall loss is as follows,

$$L_{task} = L_{pre} + \lambda L_{reg}, \tag{17}$$

where $\lambda$ is a weight to trade off the prediction and regularization losses. We use Mean Squared Error as the prediction loss, i.e., $L_{pre} = \frac{1}{\mathcal{M}}\sum_{M=1}^{\mathcal{M}}(\widehat{\mathbf{X}}_M - \mathbf{X}_M)^2$, where $\mathcal{M}$ is the training sample size, and $L_{reg}$ is $L_2$ regularization.

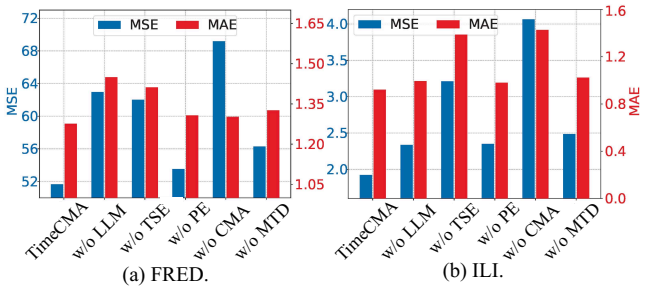
# Experiments

**Datasets.** We conduct experiments on eight datasets: ETTm1, ETTm2, ETTh1, ETTh2 (Zeng et al. 2023), ECL (Asuncion and Newman 2007), FRED-MD (McCracken and Ng 2016), ILI and Weather (Wu et al. 2021). We removed variables with missing values in the FRED-MD (Qiu et al. 2024) and simplified it as FRED.

**Baselines and Evaluation.** We evaluate seven baseline models across five categories: (1) Prompt-based LLMs: Time-LLM (Jin et al. 2024a), UniTime (Liu et al. 2024c). (2)Time series-based LLM: OFA (Zhou et al. 2023). (3) Transformer-based models: iTransformer (Liu et al. 2023), and PatchTST (Nie et al. 2023). (4) Linear-based method: Dlinear (Zeng et al. 2023). (5) CNN-based method: TimesNet (Wu et al. 2023). The evaluation metrics are mean square error (MSE) and mean absolute error (MAE). The test batch size is set to 1 for all methods to guarantee fairness during testing. Each experiment is repeated at least three times with different seeds on NVIDIA A100 GPUs.

**Main Results.** Table 1 illustrates the average performance of TimeCMA outperforms all baselines in all cases. (1) *LLM-based models perform better than deep learning and linear models.* These results verify our motivation to use LLMs for multivariate time series forecasting. (2) *Inverted embedding is essential for capturing multivariate dependencies.* For datasets with more variables, TimeCMA can perform better since we introduce inverted embedding and multivariate attention into the TimeCMA. (3) *Prompt-based LLMs outperform time series-based LLMs.* The prompt-based LLM, such as TimeCMA, outperforms the time series-based LLM, OFA, with an average improvement of 16.1% in MSE and 11.9% in MAE. This indicates that the prompt enhanced the time series embeddings. Compared to UniTime, TimeCMA shows an average improvement of about 13.9% in MSE and 12.6% in MAE.

**Ablation Studies of Model Design.** Fig. 3 indicates the ablation studies of model design, which are average values across all predictive lengths. The variant with the most significant impact is cross-modality alignment (*w/o CMA*), where CMA is replaced with concatenation. The results highlights that our similarity-based retrieval of cross-modal design is superior to simple concatenation. The next most impactful variant is the LLM. The result for *w/o LLM* signifies the LLM-empowered dual branches have better prediction results than the time series branch. Without a time series encoder (*w/o TSE*), the degradation results indicate that extracting disentangled time series embeddings is fundamental for forecasting. We find that removing the prompt encoder (*w/o PE*) has the least impact, as the LLM captures the dependencies between variables, and the prompt encoder's role is to prepare for the subsequent cross-modality alignment. Furthermore,




(a) FRED. (b) ILI.

Figure 3: Ablation study of model design.

| Models | | TimeCMA | | Time-LLM | | UniTime | | OFA | | iTransformer | | PatchTST | | TimesNet | | Dlinear | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTm1 | 96 | **0.312** | **0.351** | 0.359 | 0.381 | 0.322 | 0.363 | 0.335 | 0.369 | 0.334 | 0.368 | 0.344 | 0.373 | 0.338 | 0.375 | 0.345 | 0.372 |
| | 192 | **0.361** | **0.378** | 0.383 | 0.393 | 0.366 | 0.387 | 0.374 | 0.385 | 0.377 | 0.391 | 0.367 | 0.386 | 0.374 | 0.387 | 0.380 | 0.389 |
| | 336 | **0.392** | **0.401** | 0.416 | 0.414 | 0.398 | 0.407 | 0.407 | 0.406 | 0.426 | 0.420 | **0.392** | 0.407 | 0.410 | 0.411 | 0.413 | 0.413 |
| | 720 | 0.453 | **0.438** | 0.483 | 0.449 | **0.454** | 0.440 | 0.469 | 0.442 | 0.491 | 0.459 | 0.464 | 0.442 | 0.478 | 0.450 | 0.474 | 0.453 |
| | Avg | **0.380** | **0.392** | 0.410 | 0.409 | 0.385 | 0.399 | 0.396 | 0.401 | 0.407 | 0.410 | 0.392 | 0.402 | 0.400 | 0.406 | 0.403 | 0.407 |
| ETTm2 | 96 | **0.173** | **0.258** | 0.193 | 0.280 | 0.183 | 0.266 | 0.190 | 0.275 | 0.180 | 0.264 | 0.177 | 0.260 | 0.187 | 0.267 | 0.193 | 0.292 |
| | 192 | **0.238** | **0.301** | 0.257 | 0.318 | 0.251 | 0.310 | 0.253 | 0.313 | 0.250 | 0.309 | 0.246 | 0.305 | 0.249 | 0.309 | 0.284 | 0.362 |
| | 336 | **0.297** | **0.338** | 0.317 | 0.353 | 0.319 | 0.351 | 0.321 | 0.360 | 0.311 | 0.348 | 0.305 | 0.343 | 0.321 | 0.351 | 0.369 | 0.427 |
| | 720 | **0.393** | **0.394** | 0.419 | 0.411 | 0.420 | 0.410 | 0.411 | 0.406 | 0.412 | 0.407 | 0.410 | 0.405 | 0.408 | 0.403 | 0.554 | 0.522 |
| | Avg | **0.275** | **0.323** | 0.296 | 0.340 | 0.293 | 0.334 | 0.294 | 0.339 | 0.288 | 0.332 | 0.285 | 0.328 | 0.291 | 0.333 | 0.350 | 0.401 |
| ETTh1 | 96 | **0.373** | **0.391** | 0.398 | 0.410 | 0.397 | 0.418 | 0.398 | 0.424 | 0.386 | 0.405 | 0.404 | 0.413 | 0.384 | 0.402 | 0.386 | 0.400 |
| | 192 | **0.427** | **0.421** | 0.451 | 0.440 | 0.434 | 0.439 | 0.449 | 0.427 | 0.441 | 0.436 | 0.454 | 0.430 | 0.434 | 0.429 | 0.437 | 0.432 |
| | 336 | **0.458** | **0.448** | 0.473 | 0.451 | 0.468 | 0.457 | 0.492 | 0.466 | 0.487 | 0.458 | 0.497 | 0.462 | 0.491 | 0.469 | 0.481 | 0.459 |
| | 720 | **0.449** | **0.460** | 0.469 | 0.470 | 0.469 | 0.477 | 0.487 | 0.483 | 0.503 | 0.491 | 0.496 | 0.481 | 0.521 | 0.500 | 0.519 | 0.516 |
| | Avg | **0.423** | **0.431** | 0.448 | 0.443 | 0.442 | 0.448 | 0.457 | 0.450 | 0.454 | 0.447 | 0.463 | 0.449 | 0.458 | 0.450 | 0.456 | 0.452 |
| ETTh2 | 96 | **0.286** | **0.336** | 0.295 | 0.345 | 0.296 | 0.345 | 0.312 | 0.360 | 0.297 | 0.349 | 0.312 | 0.358 | 0.340 | 0.374 | 0.333 | 0.387 |
| | 192 | **0.363** | **0.387** | 0.386 | 0.399 | 0.374 | 0.394 | 0.387 | 0.405 | 0.380 | 0.400 | 0.397 | 0.408 | 0.402 | 0.414 | 0.477 | 0.476 |
| | 336 | **0.406** | **0.421** | 0.419 | 0.429 | 0.415 | 0.427 | 0.424 | 0.437 | 0.428 | 0.432 | 0.435 | 0.440 | 0.452 | 0.452 | 0.594 | 0.541 |
| | 720 | **0.417** | **0.438** | 0.425 | 0.442 | 0.425 | 0.444 | 0.433 | 0.453 | 0.427 | 0.445 | 0.436 | 0.449 | 0.462 | 0.468 | 0.831 | 0.657 |
| | Avg | **0.372** | **0.397** | 0.381 | 0.404 | 0.378 | 0.403 | 0.389 | 0.414 | 0.383 | 0.407 | 0.395 | 0.414 | 0.414 | 0.427 | 0.559 | 0.515 |
| ECL | 96 | **0.143** | **0.238** | 0.172 | 0.265 | 0.196 | 0.287 | 0.197 | 0.290 | 0.148 | 0.240 | 0.186 | 0.269 | 0.168 | 0.272 | 0.197 | 0.282 |
| | 192 | **0.161** | 0.259 | 0.182 | 0.279 | 0.199 | 0.291 | 0.201 | 0.292 | 0.162 | **0.253** | 0.190 | 0.273 | 0.184 | 0.289 | 0.196 | 0.285 |
| | 336 | **0.169** | **0.261** | 0.195 | 0.288 | 0.214 | 0.305 | 0.217 | 0.309 | 0.178 | 0.269 | 0.206 | 0.290 | 0.198 | 0.300 | 0.209 | 0.301 |
| | 720 | **0.219** | **0.315** | 0.233 | 0.320 | 0.254 | 0.335 | 0.253 | 0.339 | 0.225 | 0.317 | 0.247 | 0.322 | 0.220 | 0.320 | 0.245 | 0.333 |
| | Avg | **0.174** | **0.269** | 0.195 | 0.288 | 0.216 | 0.306 | 0.217 | 0.308 | 0.178 | 0.270 | 0.207 | 0.289 | 0.192 | 0.295 | 0.212 | 0.300 |
| FRED | 24 | **22.702** | **0.864** | 27.285 | 0.875 | 31.178 | 0.931 | 28.317 | 0.947 | 28.017 | 0.893 | 35.777 | 1.014 | 43.268 | 1.266 | 37.898 | 1.070 |
| | 36 | **40.880** | **1.157** | 48.730 | 1.172 | 54.172 | 1.223 | 59.520 | 1.306 | 50.837 | 1.274 | 61.034 | 1.345 | 69.514 | 1.533 | 71.047 | 1.477 |
| | 48 | **60.045** | **1.352** | 73.494 | 1.460 | 83.836 | 1.518 | 74.808 | 1.516 | 78.018 | 1.793 | 93.482 | 1.667 | 89.913 | 1.742 | 118.579 | 2.002 |
| | 60 | **65.015** | **1.509** | 108.221 | 1.758 | 118.429 | 1.830 | 83.613 | 1.641 | 90.212 | 1.693 | 133.444 | 2.011 | 116.187 | 1.976 | 156.844 | 2.221 |
| | Avg | **48.161** | **1.221** | 64.433 | 1.316 | 71.901 | 1.376 | 61.565 | 1.353 | 61.771 | 1.413 | 80.934 | 1.509 | 79.721 | 1.629 | 96.092 | 1.693 |
| ILI | 24 | **1.996** | 0.998 | 2.383 | 1.004 | 2.346 | 0.954 | 2.732 | 1.100 | 2.347 | 1.731 | 2.335 | 0.989 | 2.317 | **0.934** | 2.398 | 1.040 |
| | 36 | **1.906** | **0.915** | 2.390 | 0.993 | 1.998 | 0.912 | 2.664 | 1.063 | 2.468 | 0.998 | 2.561 | 1.035 | 1.972 | 0.920 | 2.646 | 1.088 |
| | 48 | **1.867** | **0.868** | 2.394 | 1.003 | 1.979 | 0.912 | 2.617 | 1.041 | 2.489 | 1.016 | 2.465 | 1.022 | 2.238 | 0.913 | 2.614 | 1.086 |
| | 60 | **1.920** | **0.904** | 2.562 | 1.049 | 2.109 | 0.938 | 2.478 | 1.035 | 2.471 | 1.065 | 2.189 | 0.997 | 2.027 | 0.928 | 2.804 | 1.146 |
| | Avg | **1.922** | **0.921** | 2.432 | 1.012 | 2.108 | 0.929 | 2.623 | 1.060 | 2.444 | 1.203 | 2.388 | 1.011 | 2.139 | 0.931 | 2.616 | 1.090 |
| Weather | 96 | **0.167** | **0.211** | 0.198 | 0.235 | 0.171 | 0.214 | 0.203 | 0.244 | 0.174 | 0.214 | 0.177 | 0.218 | 0.172 | 0.220 | 0.196 | 0.255 |
| | 192 | **0.212** | **0.253** | 0.240 | 0.269 | 0.217 | 0.254 | 0.247 | 0.277 | 0.221 | 0.254 | 0.222 | 0.259 | 0.219 | 0.261 | 0.237 | 0.296 |
| | 336 | **0.270** | **0.292** | 0.295 | 0.308 | 0.274 | 0.293 | 0.297 | 0.311 | 0.278 | 0.296 | 0.277 | 0.297 | 0.280 | 0.306 | 0.283 | 0.335 |
| | 720 | 0.350 | **0.348** | 0.368 | 0.353 | 0.351 | 0.343 | 0.368 | 0.356 | 0.358 | 0.349 | 0.352 | 0.347 | 0.365 | 0.359 | **0.345** | 0.381 |
| | Avg | **0.250** | **0.276** | 0.275 | 0.291 | 0.253 | **0.276** | 0.279 | 0.297 | 0.258 | 0.278 | 0.257 | 0.280 | 0.259 | 0.287 | 0.265 | 0.317 |

Table 1: Forecasting performance comparisons. The input sequence length is 36 for the Illness and FRED datasets and 96 for others.

| Dataset | ETTm1 - 96 | | | ETTm2 - 96 | | |
|---|---|---|---|---|---|---|
| Metric | Param. | Mem. | Speed | Param. | Mem. | Speed |
| Time-LLM | 44.66 | 28,882 | 1.08 | 44.95 | 29,140 | 1.08 |
| UniTime | 108.54 | 4,168 | 0.39 | 108.54 | 4,168 | 0.39 |
| OFA | 1.75 | 914 | 0.18 | 1.74 | 914 | 0.17 |
| TimeCMA | **17.99** | **821** | **0.09** | **17.99** | **818** | **0.08** |

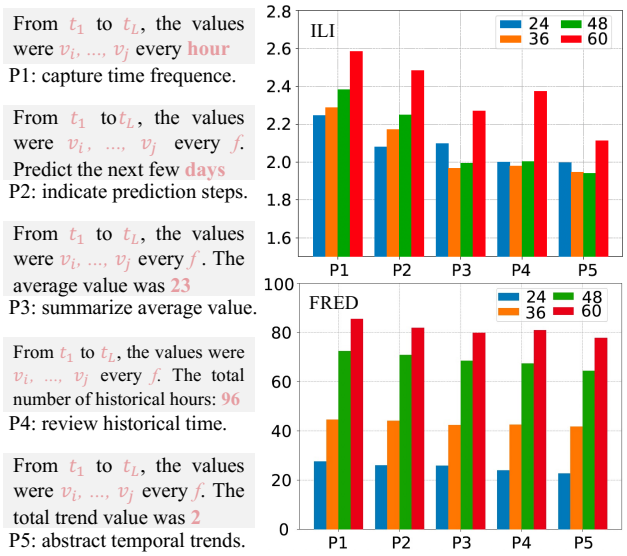
Table 2: Efficiency analysis of LLM-based baselines.

without multivariate Transformer decoder (*w/o MTD*) shows that decoding long-term temporal dependencies between multiple variables is essential for MTSF.

**Ablation Studies of Prompt Design.** We design five prompts: Prompts 1 to 5 are in Fig. 4 (a), with different intentions for the LLMs on the last token, e.g. from "to capture the frequency" to "summarize the trend". The ablation studies of prompt design are demonstrated in Fig. 4 (b) on MSE. A key insight is: *prompts where the last token is a numerical value generally have better prediction performance*, such as Prompts 3, 4, and 5. Among these numeric last-token prompts, Prompt 5 is the best since it abstracts the time series trends. The second best is prompt 3, which averages the time series but may introduce noise since the average information is not necessarily useful for forecasting. Following this is Prompt 2, which emphasizes the historical time information.

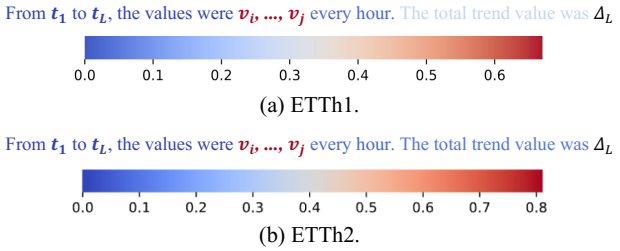
**Model Efficiency Analysis.** Table 2 provides an efficiency analysis of TimeCMA, Time-LLM, and OFA. UniTime cannot be fairly compared in terms of efficiency because it is trained on all datasets. To ensure fairness of memory, we set the training batch size to 8, thus each iteration has 8 samples. The results show that TimeCMA has the smallest training parameters and memory usage thanks to our design of the last token only and its storage. Conversely, UniTime has the largest parameters and Time-LLM has the largest memory usage and slowest speed. OFA's number of parameters, memory usage, and inference speed are second only to TimeCMA, even though it only uses time series as input data. This shows that the prompt we designed does not increase the computational costs and essentially improves the prediction.

**Last Token Attention Analysis.** We visualize the attention of the last token $< \Delta_T >$ from the final layer of GPT-2. First, we segment the words and time series values in the prompt into different segments. Then, we visualize the attention of the last token to the previous segments to verify which part of the last token receives the most attention scores. As shown in Fig. 5: the highest attention from the last token is directed toward the time series value, indicating that *the last token*



(a) Prompt design examples. (b) Results in ILI & FRED.

Figure 4: Five prompts with different purposes to trigger last token.



(a) ETTh1.

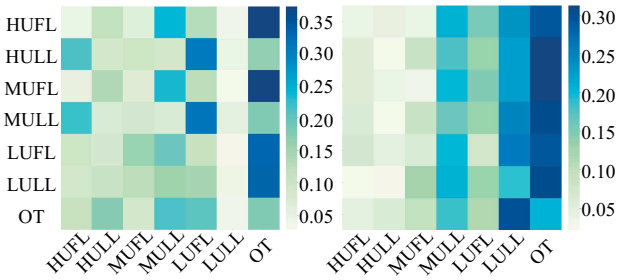
(b) ETTh2.

Figure 5: Last token attention visualization

*effectively captures the value information of the time series.*

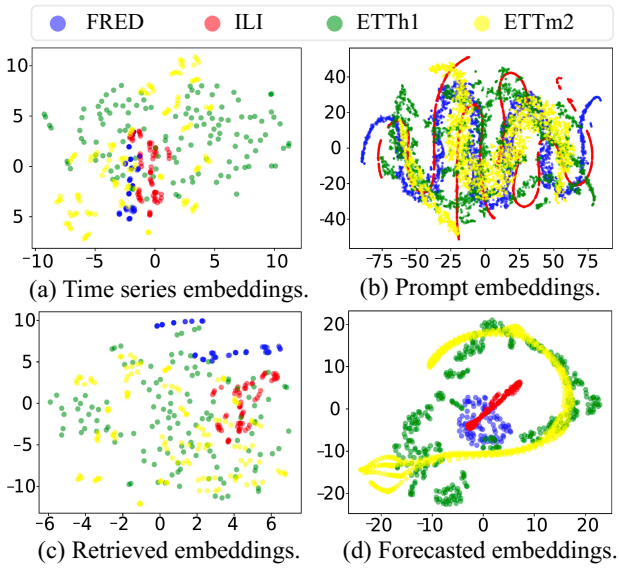
**Encoder Attention Analysis.** We visualize the variable attention map from the time series and prompt encoders, respectively, in Fig. 6 (a) and (b), each row showing its variable attention to different column variables. The time series attention is from a Pre-LN Transformer encoder, and the prompt attention is from the LLM. It shows that *Transformer and LLM capture complementary information of multivariable interrelations: the Transformer time-series attention is local and variable-specific, LLM textual attention is universal and captures global dependencies between variables.* In Fig. 6 (a), the Transformer attention map is local and captures the variable-specific temporal dependencies within the variables. In Fig. 6 (b), the LLM focuses on a broader range of variables, indicating its capability to capture global and shared dependencies effectively. Thus, integrating the LLM with the Transformer enables the TimeCMA to leverage local and global dependencies, enhancing forecasting performance.

**T-SNE Visualization.** Fig. 7 presents T-SNE visualization of time series (TS) and prompt embeddings. In Fig. 7 (a), the points are clustered by dataset, indicating that the Transformer captures the specific characteristics of each dataset.



(a) Time series encoder. (b) Prompt encoder.

Figure 6: Attention maps of Transformer and LLM encoders.



(a) Time series embeddings. (b) Prompt embeddings.

(c) Retrieved embeddings. (d) Forecasted embeddings.

Figure 7: T-SNE visualization on four datasets.

Fig. 7 (b) shows that prompt embeddings have more complex inter-relations than TS embeddings. Fig. 7 (c) tightly integrates cross-modality TS embeddings with higher similarity, making the retrieved time series embedding more cohesive. Fig. 7 (d) illustrates that forecasted TS form well-separated clusters for each dataset. This suggests that the projection effectively utilizes the retrieved embeddings to generate accurate forecasts. Overall, the step-by-step refinement shows how the TimeCMA improves data representations.

## Conclusion

This paper presents TimeCMA, an LLM-empowered framework via cross-modality alignment for multivariate time series forecasting. A cross-modality alignment module is designed to aggregate the time series and LLM branches based on channel-wise similarity retrieval to enhance forecasting. TimeCMA shows promise in using the last token embedding to reduce computational costs and accelerate the inference speed of the LLM-based method. Sufficient experiments offer insights into the efficacy and efficiency of TimeCMA.

## Acknowledgments

This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contributions from the industry partner(s).

## References

Asuncion, A.; and Newman, D. 2007. UCI machine learning repository.

BehnamGhader, P.; Adlakha, V.; Mosbach, M.; Bahdanau, D.; Chapados, N.; and Reddy, S. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv*.

Cai, J.; Wang, D.; Chen, H.; Liu, C.; and Xiao, Z. 2024. Modeling dynamic spatiotemporal user preference for location prediction: a mutually enhanced method. *WWWJ*, 27(2): 14.

Cao, D.; Jia, F.; Arik, S. O.; Pfister, T.; Zheng, Y.; Ye, W.; and Liu, Y. 2024. TEMPO: Prompt-based Generative Pre-trained Transformer for Time Series Forecasting. In *ICLR*.

Chang, C.; Chan, C.-T.; Wang, W.-Y.; Peng, W.-C.; and Chen, T.-F. 2024. TimeDRL: Disentangled Representation Learning for Multivariate Time-Series. In *ICDE*, 625–638.

Chen, H.; Wang, D.; and Liu, C. 2020. Towards Semantic Travel Behavior Prediction for Private Car Users. In *HPCC*, 950–957.

Gruver, N.; Finzi, M.; Qiu, S.; and Wilson, A. G. 2023. Large Language Models Are Zero-Shot Time Series Forecasters. In *NeurIPS*.

Huang, L.; Qin, J.; Zhou, Y.; Zhu, F.; Liu, L.; and Shao, L. 2023. Normalization Techniques in Training DNNs: Methodology, Analysis and Application. *TPAMI*, 45(8): 10173–10196.

Huang, Q.; Zhou, Z.; Yang, K.; Lin, G.; Yi, Z.; and Wang, Y. 2024. LeRet: Language-Empowered Retentive Network for Time Series Forecasting. In *IJCAI*.

Jia, F.; Wang, K.; Zheng, Y.; Cao, D.; and Liu, Y. 2024. GPT4MTS: Prompt-based Large Language Model for Multimodal Time-series Forecasting. In *AAAI*, 23343–23351.

Jin, G.; Liu, C.; Xi, Z.; Sha, H.; Liu, Y.; and Huang, J. 2022. Adaptive Dual-View WaveNet for urban spatial–temporal event prediction. *Information Sciences*, 588: 315–330.

Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J. Y.; Shi, X.; Chen, P.-Y.; Liang, Y.; Li, Y.-F.; Pan, S.; and Wen, Q. 2024a. Time-LLM: Time series forecasting by reprogramming large language models. In *ICLR*.

Jin, M.; Zhang, Y.; Chen, W.; Zhang, K.; Liang, Y.; Yang, B.; Wang, J.; Pan, S.; and Wen, Q. 2024b. Position Paper: What Can Large Language Models Tell Us about Time Series Analysis. In *ICML*.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.; and Choo, J. 2022. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In *ICLR*.

Liang, Y.; Wen, H.; Nie, Y.; Jiang, Y.; Jin, M.; Song, D.; Pan, S.; and Wen, Q. 2024. Foundation models for time series analysis: A tutorial and survey. In *KDD*.

Lin, Z.; Gou, Z.; Gong, Y.; Liu, X.; Shen, Y.; Xu, R.; Lin, C.; Yang, Y.; Jiao, J.; Duan, N.; and Chen, W. 2024. Not All Tokens Are What You Need for Pretraining. *NeurIPS*.

Liu, C.; Cai, J.; Wang, D.; Tang, J.; Wang, L.; Chen, H.; and Xiao, Z. 2021a. Understanding the regular travel behavior of private vehicles: an empirical evaluation and a semi-supervised model. *JSEN*, 21(17): 19078–19090.

Liu, C.; Wang, D.; Chen, H.; and Li, R. 2021b. Study of forecasting urban private car volumes based on multi-source heterogeneous data fusion. *Journal on Communication*, 42(3).

Liu, C.; Xiao, Z.; Long, C.; Wang, D.; Li, T.; and Jiang, H. 2024a. MVCAR: Multi-View Collaborative Graph Network for Private Car Carbon Emission Prediction. *TITS*, 1–12.

Liu, C.; Xiao, Z.; Wang, D.; Cheng, M.; Chen, H.; and Cai, J. 2022. Foreseeing private car transfer between urban regions with multiple graph-based generative adversarial networks. *WWWJ*, 25(6): 2515–2534.

Liu, C.; Xiao, Z.; Wang, D.; Wang, l.; Jiang, H.; Chen, H.; and Yu, J. 2021c. Exploiting Spatiotemporal Correlations of Arrive-Stay-Leave Behaviors for Private Car Flow Prediction. *TNSE*, 9(2): 834–847.

Liu, C.; Yang, S.; Xu, Q.; Li, Z.; Long, C.; Li, Z.; and Zhao, R. 2024b. Spatial-temporal large language model for traffic prediction. In *MDM*.

Liu, X.; Hu, J.; Li, Y.; Diao, S.; Liang, Y.; Hooi, B.; and Zimmermann, R. 2024c. UniTime: A Language-Empowered Unified Model for Cross-Domain Time Series Forecasting. In *WWW*.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2023. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *ICLR*.

Liu, Z.; Miao, H.; Zhao, Y.; Liu, C.; Zheng, K.; and Li, H. 2024d. LightTR: A Lightweight Framework for Federated Trajectory Recovery. In *ICDE*, 4422–4434.

McCracken, M. W.; and Ng, S. 2016. FRED-MD: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4): 574–589.

Miao, H.; Shen, J.; Cao, J.; Xia, J.; and Wang, S. 2022. MBA-STNet: Bayes-enhanced Discriminative Multi-task Learning for Flow Prediction. *TKDE*.

Miao, H.; Zhao, Y.; Guo, C.; Yang, B.; Kai, Z.; Huang, F.; Xie, J.; and Jensen, C. S. 2024. A unified replay-based continuous learning framework for spatio-temporal prediction on streaming data. In *ICDE*.

Nie, Y.; H. Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *ICLR*.

Niu, T.; Wang, J.; Lu, H.; Yang, W.; and Du, P. 2020. Developing a deep learning framework with two-stage feature selection for multivariate financial time series forecasting. *Expert Syst. Appl.*, 148: 113237.

Pan, Z.; Jiang, Y.; Garg, S.; Schneider, A.; Nevmyvaka, Y.; and Song, D. 2024. $S^2$IP-LLM: Semantic Space Informed Prompt Learning with LLM for Time Series Forecasting. In *ICML*.

Qiu, X.; Hu, J.; Zhou, L.; Wu, X.; Du, J.; Zhang, B.; Guo, C.; Zhou, A.; Jensen, C. S.; Sheng, Z.; and Yang, B. 2024. TFB: Towards Comprehensive and Fair Benchmarking of Time Series Forecasting Methods. In *VLDB*.

Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Smith, B. L.; and Demetsky, M. J. 1997. Traffic flow forecasting: comparison of modeling approaches. *Journal of transportation engineering*, 123(4): 261–266.

Sun, C.; Li, Y.; Li, H.; and Hong, S. 2024. TEST: Text Prototype Aligned Embedding to Activate LLM's Ability for Time Series. In *ICLR*.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *ICLR*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *AAAI*, 22419–22430.

Xiao, J.; Xiao, Z.; Wang, D.; Havyarimana, V.; Liu, C.; Zou, C.; and Wu, D. 2022. Vehicle Trajectory Interpolation Based on Ensemble Transfer Regression. *TITS*, 23(7): 7680–7691.

Xiong, R.; Yang, Y.; He, D.; Zheng, K.; Zheng, S.; Xing, C.; Zhang, H.; Lan, Y.; Wang, L.; and Liu, T. 2020. On Layer Normalization in the Transformer Architecture. In *ICML*, volume 119, 10524–10533.

Xu, R.; Miao, H.; Wang, S.; Yu, P. S.; and Wang, J. 2024. PeFAD: A Parameter-Efficient Federated Framework for Time Series Anomaly Detection. In *KDD*.

Xue, H.; and Salim, F. D. 2023. PromptCast: A New Prompt-Based Learning Paradigm for Time Series Forecasting. *TKDE*, 1–14.

Xue, H.; Voutharoja, B. P.; and Salim, F. D. 2022. Leveraging language foundation models for human mobility forecasting. In *SIGSPATIAL*, 90:1–90:9.

Yang, S.; Su, Q.; Li, Z.; Li, Z.; Mao, H.; Liu, C.; and Zhao, R. 2024. SQL-to-Schema Enhances Schema Linking in Text-to-SQL. In *DEXA*, volume 14910, 139–145.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are Transformers Effective for Time Series Forecasting? In *AAAI*.

Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI*, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *ICML*, volume 162, 27268–27286.

Zhou, T.; Niu, P.; Wang, X.; Sun, L.; and Jin, R. 2023. One Fits All: Power General Time Series Analysis by Pretrained LM. In *NeurIPS*.