Efficient Multivariate Time Series Forecasting via Calibrated Language Models with Privileged Knowledge Distillation

Chenxi Liu[1], Hao Miao[2], Qianxiong Xu[1,*], Shaowen Zhou[1], Cheng Long[1], Yan Zhao[2], Ziyue Li[3], Rui Zhao[4]

[1]*S-Lab, Nanyang Technological University, Singapore*
[2]*Aalborg University, Denmark* [3]*University of Cologne, Germany*
[4]*SenseTime Research, China*
{chenxi.liu, qianxiong.xu, shaowen.zhou, c.long}@ntu.edu.sg,
{haom, yanz}@cs.aau.dk, zlibn@wiso.uni-koeln.de, zhaorui@sensetime.com

Abstract—Multivariate time series forecasting (MTSF) endeavors to predict future observations given historical data, playing a crucial role in time series data management systems. With advancements in large language models (LLMs), recent studies employ textual prompt tuning to infuse the knowledge of LLMs into MTSF. However, the deployment of LLMs often suffers from low efficiency during the inference phase. To address this problem, we introduce TimeKD, an efficient MTSF framework that leverages the calibrated language models and privileged knowledge distillation. TimeKD aims to generate high-quality future representations from the proposed cross-modality teacher model and cultivate an effective student model. The cross-modality teacher model adopts calibrated language models (CLMs) with ground truth prompts, motivated by the paradigm of Learning Under Privileged Information (LUPI). In addition, we design a subtractive cross attention (SCA) mechanism to refine these representations. To cultivate an effective student model, we propose an innovative privileged knowledge distillation (PKD) mechanism including correlation and feature distillation. PKD enables the student to replicate the teacher's behavior while minimizing their output discrepancy. Extensive experiments on real data offer insight into the effectiveness, efficiency, and scalability of the proposed TimeKD.

Index Terms—Multivariate Time Series, Privileged Knowledge Distillation, Large Language Models

I. INTRODUCTION

The rapid revolution of sensing techniques and the proliferation of edge devices generate massive amounts of time series data [1]–[4]. It is important to extract useful knowledge from the historical time series data enabling a variety of time series-related applications, such as arrival time estimation [5]–[7], traffic prediction [8]–[12], and route planning [13]–[15]. A fundamental aspect of such applications is multivariate time series forecasting (MTSF), which aims to predict future observations given historical data [16]–[18]. MTSF serves as an essential functionality in the time series data management systems [19]–[24].

Due to its significance, substantial research has been devoted to inventing effective MTSF models [25]. Generally, existing MTSF methods can be categorized into classical

methods [26]–[29] and large language model (LLM)-based methods [30]–[33]. Small classical methods often adopt convolutional neural networks (CNNs) [26], fully-connected networks (FC) [27], and Transformers [28] to learn spatial and temporal correlations across variables and time steps for time series forecasting [34]. However, the limited number of learnable parameters and small-scale training datasets lead these classic methods to extract only shallow features. Recent studies apply large language models for time series forecasting inspired by the fact that time series and language data exhibit similar sequential formats, as well as the generic pre-trained knowledge and knowledge transfer capabilities of LLMs [35], [36]. These methods often consider LLMs as deep feature extractors to capture deep features from historical time series facilitating effective time series forecasting.

Unfortunately, due to the huge amount of parameters, the deployment of LLMs (e.g., LLaMA-7B [37]) would *deteriorate the inference efficiency*, posing challenges to real-time time series forecasting. For example, Time-LLM [38], built on LLaMA-7B, uses 96 historical time steps to predict the next 96 on the ETTm1 dataset, corresponding inference time is 0.135 seconds per sample, 13 times longer than a small classic model, iTransformer [29]. To address this problem, we propose to adopt knowledge distillation (KD) [39] to distill the robust representation capability of large-scale LLMs (teacher) to a small-scale classical model (student), so that the learned student model enables both **deep feature extraction** and **fast inference**. Nevertheless, it is non-trivial to develop LLM-guided KD (LLM-KD) methods for MTSF due to the following challenges.

Challenge I: High-quality Teacher Model Training. The intuitive approach to designing the teacher model for MTSF tasks is to leverage LLMs to predict future time series, as illustrated in the upper part of Figure 1. Specifically, these methods employ traditional teacher models, which take history data as input and produce future features as output [40]. However, this limits the predictive capability of the teacher model due to the constrained distribution of historical data. In addition, the student model may inherit the prediction bias of
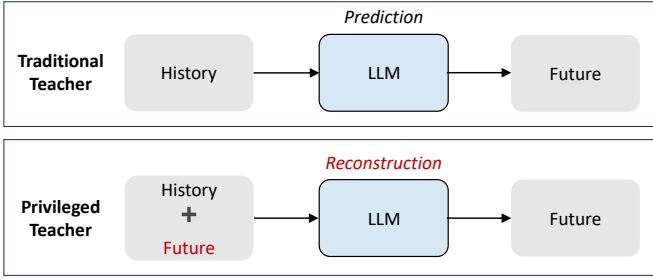
Fig. 1: Comparison of traditional teacher models and our privileged teacher. Future data is available only during training and not during testing, thus regarded as privileged information.

the teacher model during training. This is particularly evident when dealing with out-of-distribution data [41], where the student model's generalization ability may be weaker.

Challenge II: Effective Student Model Cultivation. Prevailing LLM-KD methods typically optimize the student model with the output discrepancy of the teacher model. For example, existing LLM-KD methods often rely on black-box distillation [42], in which the student model only has access to the outputs of the teacher model. These approaches fail to fully capture the teacher model's behavior during the training phase, leading to incomplete knowledge transfer. Consequently, it is crucial to develop knowledge distillation methods that transfer the training behaviors and outputs from the teacher model to foster effective student model cultivation.

This study addresses the above challenges by providing an efficient multivariate time series forecasting framework via calibrated language models with privileged knowledge distillation, entitled TimeKD. To train a high-quality teacher model (*Challenge I*), we develop a cross-modality teacher model for time series reconstruction. Existing methods [33], [43] demonstrate that wrapping time series as textual prompts enables effective time series feature extraction because the trained LLMs cannot understand pure time series data due to the lack of textual instructions. Building on this insight, We pre-define a prompt template according to the original time series with additional contextual information, e.g., "*The values were 10, 11, and 20 every hour. Forecast the values for the next 24 hours*".

Moreover, we leverage the Learning Under Privileged Information (LUPI) [44] paradigm in the teacher model. In particular, we treat future data (i.e., ground truth prompts) as privileged information and directly input them into the LLMs to extract effective future representations, as illustrated in the lower part of Figure 1. It is notable that taking ground truth as teacher LLM's inputs is feasible for the KD framework without data leakage since the teacher LLMs are only used during training and not during testing. Further, we design calibrated language models (CLMs) to purify the representations of the prompt using a novel calibrated attention mechanism: this mechanism suppresses the inter-modality fusion while enhancing the intra-modality correlations, e.g., the attention scores between text and time series tokens would be biased with a

negative value. Subsequently, we design a subtractive cross-attention (SCA) mechanism to remove the textual information doped in learned future time series representations.

To cultivate an effective student model for time series forecasting (*Challenge II*), we propose an innovative privileged knowledge distillation (PKD). PKD focuses on both correlation and feature distillations to transfer the privileged representations from the LLM-empowered teacher model to a lightweight student model. In terms of correlation distillation, the student model replicates the teacher's behavior based on the shared attention maps of teacher and student models. In terms of feature distillation, the teacher cultivates the student model by minimizing the output discrepancy. By combining these two complementary distillation methods, the teacher model effectively guides the student model in learning internal behavior and enhancing output performance, enabling efficient and effective time series forecasting.

Our primary contributions are summarized as follows.

- To the best of our knowledge, this is the first systematic study to leverage privileged knowledge distillation for time series forecasting. We propose TimeKD, an effective and efficient time series forecasting framework with calibrated language models.
- We develop a cross-modality teacher model comprising the calibrated language models and the subtractive cross attention mechanism, facilitating effective future time series representation extraction.
- We propose the privileged knowledge distillation, which includes correlation and feature distillations leveraging privileged information, enables the student model to learn the teacher's behavior while minimizing the output discrepancy between them.
- We report on experiments on real data, offering evidence of the effectiveness, efficiency, and scalability of the proposed TimeKD.

The remainder of this paper is structured as follows. Section II surveys related work. Section III covers preliminary concepts and formalizes the problem of MTSF. We detail the TimeKD framework in Section IV, followed by the experimental study in Section V, and Section VI concludes the paper.

II. RELATED WORK

We briefly review prior studies on LLMs-based time series forecasting and LLMs-based knowledge distillation.

A. LLMs-based Time-Series Forecasting

Time series forecasting, which predicts events based on sequential temporal data, has gained increasing attention over the past decades [45]–[47]. Recently, LLMs-based time series forecasting methods [30]–[32], [43], [48] have attracted increasing interest due to the powerful knowledge transfer capability of LLMs and the increasing availability of time series data. These models aim to harness the cross-domain knowledge and the transfer learning capabilities of LLMs for temporal correlation capturing. In particular, recent studies, e.g., TEMPO [48], OFA [30], LLM4TS [31], and TEST [32]

perform time series forecasting by fine-tuning. These models have integrated different mechanisms to adapt the LLMs to capture a variety of temporal semantics facilitating time series forecasting. Further, LLM-based methods utilize multiple data modalities to enhance time series forecasting, for example, introducing prompts with text as additional inputs enabling effective time series feature extraction.

Existing multimodal LLMs for time series modeling methods can be divided into channel-independent time series forecasting models [33], [38], [49]–[51] and channel-dependent time series forecasting models [43], [52]. Channel-independent time series forecasting methods [33], [38], [49] often treat each time series variable separately. Time-LLM reprograms an LLM for time series forecasting where the backbone language model remains intact [38]. UniTime introduces a unified model that leverages natural language prompts and a Language-TS Transformer to align domain-specific characteristics for effective cross-domain time series forecasting [49]. However, these models often rely on channel independence, overlooking dependencies between multiple variables, resulting in longer training times and suboptimal performance. Channel-dependent time series forecasting methods [43] aims to extract effective features by learning correlations across multiple variables in time series based on LLMs. TimeCMA [43] focuses on handling data entanglement through cross-modality alignment. However, it remains a key problem to develop lightweight, efficient, and scalable LLMs-based time series forecasting methods.

B. LLMs-based Knowledge Distillation

Knowledge distillation (KD) can generally be categorized into black-box distillation [42] and white-box distillation [53]. In black-box KD, the student model has access only to the teacher model's predictions, while in white-box KD, the student can directly leverage the internal weights of the teacher model [54]. With the rapid advancements in large language models (LLMs), black-box distillation KD has emerged as an effective technique to leverage cross-domain knowledge and mitigate the high computational costs. These LLM-based KD have been applied for various tasks, such as language generation [55], [56], graph learning [57], and recommendation [58].

Recently, LLMs-based knowledge distillation [40], [42] has been employed in time series. For example, Anomaly-LLM [42], introduces a knowledge distillation framework for time series anomaly detection, where the student model is trained to replicate the output features of an LLM. This approach enhances the training process by augmenting the time series data and inputting these samples into a partially frozen LLM. However, LLMs are not inherently designed for time series data, as they are pre-trained on extensive language corpora rather than numerical sequences. Moreover, the linear transformations involved in the anomaly detection would obscure information embedded in the teacher model's parameters. As a result, relying solely on the LLM's final features for training the student model is insufficient to capture the knowledge required for downstream tasks.

From *<t-H+1>* to *<t>*, values were *<h_i, ..., h_j>* every *<f>* minutes. Next *<M>* minutes: *<g_i, ..., g_j>*

(a) Ground truth and historical prompt.

From *<t-H+1>* to *<t>*, values were *<h_i, ..., h_j>* every *<f>* minutes. Forecast the next *<M>* minutes

(b) Historical prompt.

Fig. 2: Examples of input prompts

III. PROBLEM STATEMENT

We proceed to present the necessary preliminaries and then define the problem addressed.

Definition 1 (Multivariate Time Series). The multivariate time series observations, denoted as $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|X|}\} \in \mathbb{R}^{|X| \times N}$, is a time ordered sequence of observations, where N is the number of variables. Each observation $\mathbf{x}_i$ is a N-dimensional vector indicating N features (e.g., load or temperature) at time step i. The term v_i refers to the value of the time series at time step i. Notably, we define three types of multivariate time series: the historical time series $\mathbf{X}_H \in \mathbb{R}^{H \times N}$, the Ground truth $\mathbf{X}_G \in \mathbb{R}^{G \times N}$, and the observed time series $\mathbf{X}_O \in \mathbb{R}^{O \times N}$. The sequences $\mathbf{X}_H$ and $\mathbf{X}_G$ are employed during the training stage to develop a high-quality teacher model, while $\mathbf{X}_O$ is utilized in the test stage for forecasting.

Definition 2 (Prompt). The prompt integrates time series values in the text format with a pre-defined template [33], [43], word instructions about the time series forecasting task (please see Figure 2). Specifically, the historical time series $\mathbf{X}_H \in \mathbb{R}^{H \times N}$ are transformed into a set of textual prompt $\mathbf{P}_{\text{HD}} \in \mathbb{R}^{W_{\text{HD}} \times N}$, where W_{HD} indicates the number of words in each historical data prompt, corresponding to the H steps of time series. HD denotes historical data.

Multivariate Time Series Forecasting. Given a time series with O observations $\mathbf{X}_O \in \mathbb{R}^{O \times N}$, the objective is to forecast future time series $\mathbf{X}_M \in \mathbb{R}^{M \times N}$ over M time steps through a student model $f(\cdot)$ learned from a teacher model (i.e., LLMs) based on privileged knowledge distillation. Specifically, $f(\cdot)$ is trained using the historical time series $\mathbf{X}_H$ along with its textual prompt $\mathbf{P}_{\text{HD}}$, as well as ground truth $\mathbf{X}_G$ and its corresponding prompt $\mathbf{P}_{\text{GT}}$.

IV. METHODOLOGY

In this section, we introduce TimeKD, an efficient multivariate <u>time</u> series forecasting framework via calibrated language models with privileged <u>k</u>nowledge <u>d</u>istillation. We first give an overview of the framework and then provide specifics.
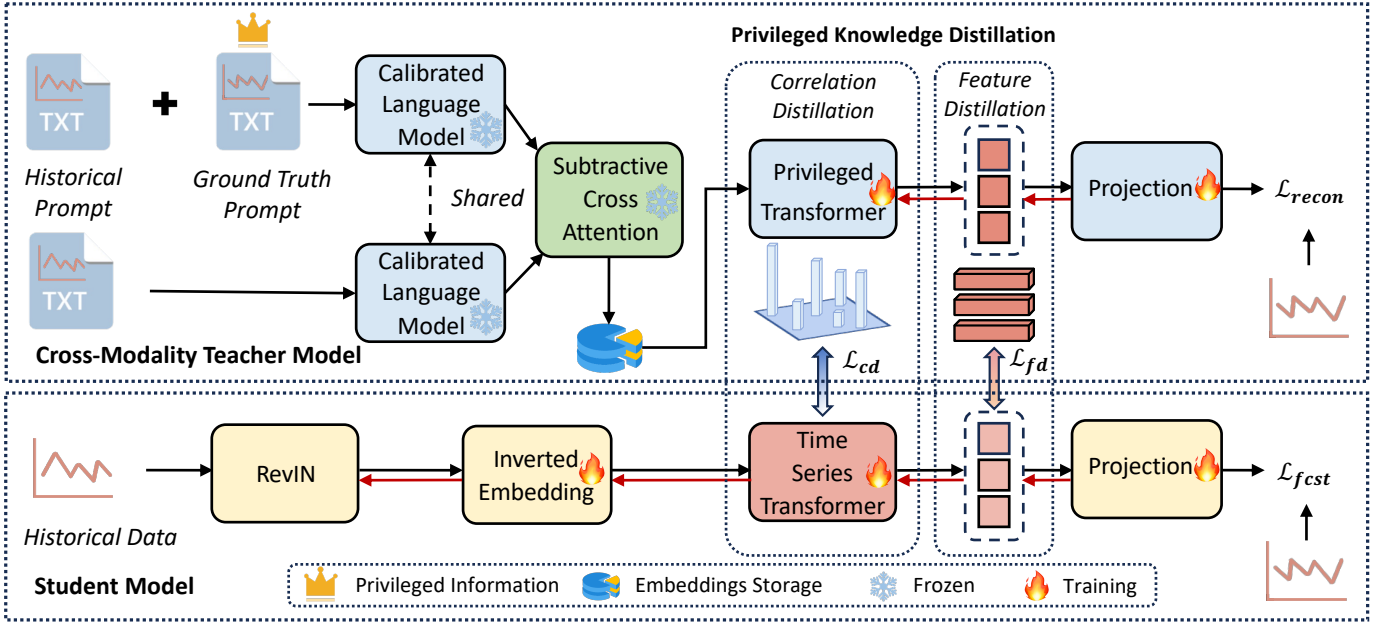
Fig. 3: TimeKD Framework. **Cross-Modality Teacher Model** processes textual prompts to reconstruct time series during the training stage. **Student Model** learned from the teacher model via privileged knowledge distillation for efficient forecasting.

A. Overall Framework

As illustrated in Figure 3, TimeKD comprised a cross-modality teacher model and a student model with privileged knowledge distillation used to transfer knowledge from the teacher model and thereby train a powerful student model.

Cross-Modality Teacher Model. This model mainly consists of calibrated language models (CLMs), a subtractive cross attention (SCA), and a privileged Transformer $PTEncoder(\cdot)$ for the reconstruction task. The ground truth $\mathbf{P}_{GT}$ and historical data $\mathbf{P}_{HD}$ prompts as privileged information, which are input to the CLMs respectively, to facilitate effective future representation generation. The SCA is designed to remove textual information doped in the future time series representations. Then, we input these features into the $PTEncoder(\cdot)$, a lightweight Pre-LN Transformer encoder [59], to reconstruct the time series ground truth $\mathbf{X}_G$.

- **Calibrated Language Models.** This module generates future time-series features through a combination of components: a tokenizer, layer normalizations, calibrated attention mechanisms, and feed-forward networks. The calibrated attention mechanisms are specifically designed to ensure modality consistency between textual and time-series tokens. The last token embeddings are extracted for efficient knowledge distillation.
- **Subtractive Cross Attention.** The module removes textual information doped in the last token embeddings, ensuring that the retained features are highly relevant for time series. These features are stored and utilized as privileged information for cultivating the student model, reducing the computational costs.

Student Model. This model processes historical data $\mathbf{X}_H$ through a reversible instance normalization layer (RevIN) [60], followed by an inverted embedding. The inverted embedding layer embeds the whole time series of each variate independently. Then, a time series Transformer $TSTEncoder(\cdot)$, which is a lightweight Transformer with the same structure as $PTEncoder(\cdot)$, processesl these embeddings to capture long-term temporal dependencies across multiple variables for forecasting the time series ground truth $\mathbf{X}_G$.

Privileged Knowledge Distillation. This module transfers the future representations from the teacher to the student model through two losses: correlation distillation loss $\mathcal{L}_{cd}$ and feature distillation loss $\mathcal{L}_{fd}$. The $\mathcal{L}_{cd}$ aligns the correlations between the Transformers from the teacher and the student model, leveraging the student model to imitate the teacher's behavior. Meanwhile, the $\mathcal{L}_{fd}$ minimizes the output discrepancy between teacher and student models.

Finally, the observed time series $\mathbf{X}_O$ to used to predict the future time series $\mathbf{X}_M$ via a well-learned student model.

B. Cross-Modality Teacher Model

We train an LLM-based cross-modality teacher model to reconstruct the time series ground truth for learning high-quality future representations.

1) Calibrated Language Models: Calibrated language models (CLMs) modify the masked multi-self attention in large language models (LLMs) based on the cross-modality and intra-modality correlations of prompt tokens. The CLMs are composed of a tokenizer, layer normalization layers, calibrated attention mechanisms, feed-forward networks, and a last token extractor, as shown in Figure 4.
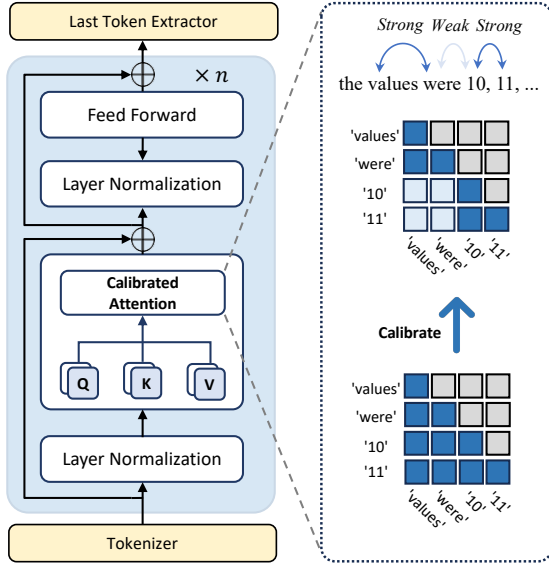
Fig. 4: An example of calibrated attention score in the CLMs.

Given the historical data prompt $\mathbf{P}_{\text{HD}} = \{\mathbf{p}_1, \ldots, \mathbf{p}_{W_{\text{HD}}}\} \in \mathbb{R}^{W_{\text{HD}} \times N}$ and ground truth prompt $\mathbf{P}_{\text{GT}} = \{\mathbf{p}_1, \ldots, \mathbf{p}_{W_{\text{GT}}}\} \in \mathbb{R}^{W_{\text{GT}} \times N}$, where W_{HD} and W_{GT} are the words in each prompt, and $W_{\text{HD}} < W_{\text{GT}}$. We first input them into the CLMs separately to get the prompt embeddings. The tokenizer is responsible for converting these prompts into a series of token IDs $\mathbf{I}_{\text{GT}} \in \mathbb{R}^{S_{\text{HD}} \times N}$ and $\mathbf{I}_{\text{HD}} \in \mathbb{R}^{S_{\text{GT}} \times N}$, where S_{HD} and S_{GT} represents the number of token ID.

Subsequently, these tokenized prompt representations are processed by the CLMs to generate contextualized embeddings. This involves a series of transformations, including calibrated attention, layer normalization, and feed-forward operations, which progressively refine the token representations across multiple layers:

$$\overline{\mathcal{I}}_{\text{GT}}^i = CalAtt(LN(\mathbf{I}_{\text{GT}}^i)) + \mathbf{I}_{\text{GT}}^i, \tag{1}$$

$$\mathcal{I}_{\text{GT}}^{i+1} = FFN(LN(\overline{\mathcal{I}}_{\text{GT}}^i)) + \overline{\mathcal{I}}_{\text{GT}}^i, \tag{2}$$

where $\overline{\mathcal{I}}_{\text{GT}}^i \in \mathbb{R}^{S_{\text{HD}} \times N \times D}$ represents the intermediate representation of the i_{th} layer after applying the $CalAtt(\cdot)$ and the $LN(\cdot)$. D denotes the hidden dimension of the language model. $\mathbf{I}_{\text{GT}}^0 = [\mathbf{I}_{\text{GT}} + \mathbf{PE}]$, where $\mathbf{PE}$ represents the learnable positional encoding.

Then, we design the calibrated attention mechanism $CalAtt$ to enhance the masked multi-self attention (MMSA) within LLMs for processing multi-modality data, such as time series and text. Traditional MMSA often struggles to distinguish the significance of cross-modality and intra-modality interactions, resulting in data entanglement issues [43]. For instance, as shown in Figure 4, the original attention mask's attention scores (bottom) display a uniform distribution across the lower triangular region. In contrast, the calibrated attention mechanism (top) strengthens intra-modality interactions by reducing the weights of cross-modality interactions (e.g., be-

tween the time series token '10' and the text token 'were'). The formulation of $CalAtt$ is as follows.

$$CalAtt(\tilde{\mathbf{I}}_{\text{GT}}^i) = \xi_o \left(Att(\xi_q \tilde{\mathbf{I}}_{\text{GT}}^i, \xi_k \tilde{\mathbf{I}}_{\text{GT}}^i, \xi_v \tilde{\mathbf{I}}_{\text{GT}}^i) \right), \tag{3}$$

$$Att(\tilde{\mathbf{I}}_{\text{GT}}^i, \tilde{\mathbf{I}}_{\text{GT}}^i, \tilde{\mathbf{I}}_{\text{GT}}^i) = \text{softmax} \left(\frac{\tilde{\mathbf{I}}_{\text{GT}}^i \tilde{\mathbf{I}}_{\text{GT}}^i{}^\top}{\sqrt{d_k}} + \text{Mask} \right) \tilde{\mathbf{I}}_{\text{GT}}^i, \tag{4}$$

$$\text{Mask}[i, j] = \begin{cases} -\Delta, & \text{if tokens } i \text{ and } j \text{ are cross-modality}, \\ 0, & \text{if tokens } i, j \text{ are intra-modality}, \end{cases} \tag{5}$$

where $\tilde{\mathbf{I}}_{\text{GT}}^i$ is the output of $\mathbf{I}_{\text{GT}}^i$ after passing through the first LN. $Att(\tilde{\mathbf{I}}_{\text{GT}}^i, \tilde{\mathbf{I}}_{\text{GT}}^i, \tilde{\mathbf{I}}_{\text{GT}}^i)$ represents the attention mechanism that computes the attention-weighted combination of values $\tilde{\mathbf{I}}_{\text{GT}}^i$ using the query $\tilde{\mathbf{I}}_{\text{GT}}^i$ and key $\tilde{\mathbf{I}}_{\text{GT}}^i$, which are derived from the input $\tilde{\mathbf{I}}_{\text{GT}}^i$ through the linear transformations ξ_q, ξ_k, and ξ_v, respectively. The term d_k represents the dimensionality of the key vectors, and it is used to scale the dot-product attention scores for numerical stability. The term Mask adjusts the attention scores by penalizing cross-modality interactions.

The LN and FFN in the CLMs are defined as follows:

$$LN\left(\mathbf{I}_{\text{GT}}^i\right) = \gamma \odot \frac{\mathbf{I}_{\text{GT}}^i - \mu}{\sigma} + \beta, \tag{6}$$

$$FFN(\hat{\mathbf{I}}_{\text{GT}}^i) = \max(0, \mathbf{W}_1 \hat{\mathbf{I}}_{\text{GT}}^i + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{7}$$

where $\hat{\mathbf{I}}_{\text{GT}}^i$ is the output of $\mathbf{I}_{\text{GT}}^i$ after the second LN. γ and β are learnable scaling and translation parameters. μ and σ represent the mean and standard deviation, respectively. $\odot$ denotes element-wise multiplication.

The outputs of the CLMs are denoted as $\mathcal{I}_{\text{GT}} \in \mathbb{R}^{S_{\text{GT}} \times N \times D}$ and $\mathcal{I}_{HD} \in \mathbb{R}^{S_{\text{HD}} \times N \times D}$. We then extract the last token embeddings from these outputs, which is motivated by the observation that the last token in a prompt encapsulates the most comprehensive knowledge due to the masked attention mechanism in LLMs. Specifically, the representation of the last token at a given position is influenced solely by the representations of its preceding tokens. As illustrated in Figure 4, assuming '11' is the last token in a prompt, it can attend to the token '10' (highlighted by the blue cubes). However, token '10' cannot attend to '11' due to the masking mechanism.

To leverage this property and reduce computational overhead for efficient knowledge distillation, we extract the last token embeddings: $\mathbf{L}_{\text{GT}} = \{\mathbf{l}_1, \ldots, \mathbf{l}_N\} \in \mathbb{R}^{N \times D}$ from the $\mathcal{I}_{\text{GT}} \in \mathbb{R}^{S_{\text{HD}} \times N \times D}$ and $\mathbf{L}_{\text{HD}} = \{\mathbf{l}_1, \ldots, \mathbf{l}_N\} \in \mathbb{R}^{N \times D}$ from the $\mathcal{I}_{\text{HD}} \in \mathbb{R}^{S_{\text{GT}} \times N \times D}$. This strategy ensures computational efficiency while preserving the essential information for subsequent distillation.

2) Subtractive Cross Attention: We design a subtractive cross-attention (SCA) mechanism to eliminate textual information embedded in the last token representations, ensuring that the retained embedding remains highly relevant to time series forecasting.

In SCA, we begin by applying layer normalization and projection functions, denoted as φ_q, φ_k, and φ_v, to the projected embeddings of the ground truth $\mathbf{L}_{\text{GT}}$ and historical
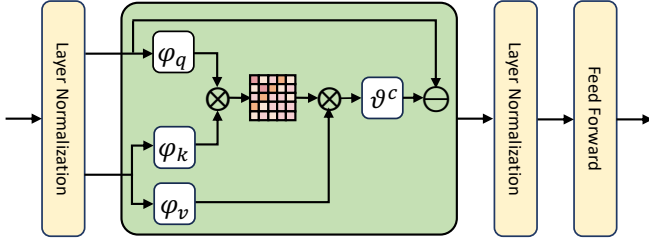
Fig. 5: Subtractive cross attention.

data $\mathbf{L}_{\text{HD}}$. We then compute the channel-wise similarity matrix $\mathbf{M}_C \in \mathbb{R}^{D \times D}$ via matrix multiplication, followed by a softmax operation:

$$\mathbf{M}_C = \text{softmax}\left(LN(\varphi_q(\mathbf{L}_{\text{GT}})) \otimes LN\varphi_k(\mathbf{L}_{\text{HD}})\right), \quad (8)$$

where $\otimes$ denotes matrix multiplication.

Next, we perform channel-wise feature aggregation by multiplying $\varphi_v(\mathbf{L}_{\text{HD}})$ with $\mathbf{M}_C$. The refined ground truth prompt embedding, $\overline{\mathbf{L}}_{\text{GT}}$, is then derived by subtracting this intersection from both $\mathbf{L}_{\text{HD}}$ and $\mathbf{L}_{\text{GT}}$, followed by layer normalization and a feed-forward layer:

$$\overline{\mathbf{L}}_{\text{GT}} = FFN(LN(\mathbf{L}_{\text{GT}} \ominus \vartheta^c \left(\varphi_v\left(\mathbf{L}_{\text{HD}}\right) \otimes \mathbf{M}_C\right))), \quad (9)$$

where $\overline{\mathbf{L}}_{\text{GT}} \in \mathbb{R}^{N \times D}$ represents the refined ground truth embedding. Here, ϑ^c is a linear layer, and $\ominus$ denotes the subtraction operation.

The SCA refines the ground truth prompt embedding, preserving privileged knowledge for further distillation. Additionally, to avoid repetitive processing with the frozen CLMs, we store the subtracted embeddings for efficient reconstruction.

3) Cross-Modality Reconstruction: This task leverages a Transformer Encoder $PTEncoder(\cdot)$ to reconstruct the ground truth in the time series modality using $\overline{\mathbf{L}}_{\text{GT}}$ from the textual modality.

Inside $PTEncoder(\cdot)$, the $\overline{\mathbf{L}}_{\text{GT}}$ first undergo layer normalization $LN(\cdot)$ at the i_{th} layer:

$$\widetilde{\mathbf{L}}_{\text{GT}}^i = LN(\overline{\mathbf{L}}_{\text{GT}}), \quad (10)$$

where $\widetilde{\mathbf{L}}_{\text{GT}}^i$ represents the intermediate embedding after normalization. The structure of $LN(\cdot)$ is the same as in Equation (6).

The normalized embeddings $\widetilde{\mathbf{L}}_{\text{GT}}^i$ are then passed through a multi-head attention layer within $PTEncoder(\cdot)$, denoted as $PTAtt(\cdot)$. The output, $\check{\mathbf{L}}_{\text{GT}}^i$, is combined with the input through a residual connection:

$$\check{\mathbf{L}}_{\text{GT}}^i = PTAtt(\widetilde{\mathbf{L}}_{\text{GT}}^i) + \overline{\mathbf{L}}_{\text{GT}}^i, \quad (11)$$

$$PTAtt(\widetilde{\mathbf{L}}_{\text{GT}}^i) = \zeta_o\left(Att(\zeta_q\widetilde{\mathbf{L}}_{\text{GT}}^i, \zeta_k\widetilde{\mathbf{L}}_{\text{GT}}^i, \zeta_v\widetilde{\mathbf{L}}_{\text{GT}}^i)\right), \quad (12)$$

$$Att(\widetilde{\mathbf{L}}_{\text{GT}}^i) = softmax\left(\frac{\widetilde{\mathbf{L}}_{\text{GT}}^i\widetilde{\mathbf{L}}_{\text{GT}}^{i\top}}{\sqrt{d_k}}\right)\widetilde{\mathbf{L}}_{\text{GT}}^i, \quad (13)$$

where ζ_o, ζ_q, ζ_k, and ζ_v are learnable linear projections. The attention computes dependencies across feature dimensions.

Algorithm 1 Cross-Modality Teacher Model Training

Input: Ground truth with historical prompts $\mathbf{P}_{\text{GT}}$ and historical prompts $\mathbf{P}_{\text{HD}}$
Output: Privileged Transformer attentions $\mathbf{A}_{\text{PE}}$, privileged embeddings $\mathbf{E}_{\text{GT}}$, and reconstructed time series $\hat{\mathbf{X}}_L$

1: **while** not converged **do**
2:    $\mathbf{L}_{\text{HD}}, \mathbf{L}_{\text{GT}} \leftarrow CLM(\mathbf{P}_{\text{HD}}), CLM(\mathbf{P}_{\text{GT}})$
3:    $\overline{\mathbf{L}}_{\text{GT}} \leftarrow SCA(\mathbf{L}_{\text{GT}}, \mathbf{L}_{\text{HD}})$
4:    $\mathbf{E}_{\text{GT}}, \mathbf{A}_{\text{PE}} \leftarrow PTEncoder(\overline{\mathbf{L}}_{\text{GT}})$
5:    $\hat{\mathbf{X}}_G \leftarrow \mathbf{W}_l\mathbf{E}_{\text{GT}} + \mathbf{b}_l$
6:    $\mathcal{L}_{\text{recon}} \leftarrow \frac{1}{\text{L}} \sum_{L=1}^{\text{L}} SL1(\hat{\mathbf{X}}_G - \mathbf{X}_G)$
7:    Update model parameters using $\mathcal{L}_{\text{recon}}$
8: **end while**
9: **Return:** $\mathbf{A}_{\text{PE}}, \mathbf{E}_{\text{GT}}, \hat{\mathbf{X}}_G$

The output $\check{\mathbf{L}}_{\text{GT}}^i$ is then normalized again through another $LN(\cdot)$, followed by a feed-forward network $FFN(\cdot)$. The result is combined with the input via a residual connection:

$$\mathbf{E}_{\text{GT}}^{i+1} = FFN(LN(\check{\mathbf{L}}_{\text{GT}}^i)) + \check{\mathbf{L}}_{\text{GT}}^i, \quad (14)$$

where $\mathbf{E}_{\text{GT}}^{i+1} \in \mathbb{R}^{D \times N}$ represents the output of $PTEncoder(\cdot)$, denoted as $\mathbf{E}_{\text{GT}}$ for simplify, and the structure of $FFN(\cdot)$ is identical to Equation (7).

The reconstructed time series is then generated by applying a projection function to $\mathbf{E}_{\text{GT}}$:

$$\hat{\mathbf{X}}_G = \mathbf{W}_l\mathbf{E}_{\text{GT}} + \mathbf{b}_l, \quad (15)$$

where $\hat{\mathbf{X}}_G \in \mathbb{R}^{G \times N}$ is the reconstructed time series, and $\mathbf{W}_l$ and $\mathbf{b}_l$ are learnable parameters.

The reconstruction loss is defined using SmoothL1 loss $(SL1)$:

$$\mathcal{L}_{\text{recon}} = \frac{1}{\text{G}} \sum_{G=1}^{\text{G}} SL1(\hat{\mathbf{X}}_G - \mathbf{X}_G), \quad (16)$$

$$SL1(\hat{\mathbf{X}}_G - \mathbf{X}_G) = \begin{cases} 0.5(\hat{\mathbf{X}}_G - \mathbf{X}_G)^2, & \text{if } |\hat{\mathbf{X}}_G - \mathbf{X}_G| < 1, \\ |\hat{\mathbf{X}}_G - \mathbf{X}_G| - 0.5, & \text{otherwise,} \end{cases} \quad (17)$$

where L is the total number of reconstruction samples. The SmoothL1 loss ensures robustness to outliers while maintaining sensitivity to smaller errors.

C. Student Model

The student model processes historical time series through the RevIN, an inverted embedding layer, a time series Transformer, and a projection layer. Given historical data $\mathbf{X}_H \in \mathbb{R}^{H \times N}$, the inverted embedding transforms $\mathbf{X}_H$ into learnable matrices $\mathbf{I}_H \in \mathbb{R}^{H \times N}$ to capture temporal dependencies across multiple variables [29]. The $\mathbf{X}_O$ is first normalized via RevIN to mitigate distribution shifts. The normalized data is then embedded as follows:

$$\mathbf{I}_H = \mathbf{W}_i\mathbf{X}_H + \mathbf{b}_i, \quad (18)$$

where $\mathbf{I}_H = \{\mathbf{i}_1, \ldots, \mathbf{i}_T\} \in \mathbb{R}^{D \times N}$ is the output of the inverted embedding layer. C denotes the hidden dimension of the embeddings, while $\mathbf{W}_i$ and $\mathbf{b}_i$ are learnable parameters.

The embeddings $\mathbf{I}_H$ are then passed into a Transformer encoder $TSTEncoder(\cdot)$, which models the temporal dependencies across multiple variables. The time series encoder consists of multiple layers, each with the following pipeline:

$$\widetilde{\mathbf{T}}_H^i = LN(\mathbf{I}_H), \tag{19}$$

$$\overline{\mathbf{T}}_H^i = TSTAtt(\widetilde{\mathbf{T}}_H^i) + \mathbf{T}_H^i, \tag{20}$$

$$\overline{\mathbf{T}}_H^{i+1} = FFN(LN(\dot{\mathbf{T}}_H^i)) + \overline{\mathbf{T}}_H^i, \tag{21}$$

where $\widetilde{\mathbf{T}}_H^i$ represents the intermediate embedding after layer normalization and $\overline{\mathbf{T}}_H^i$ is the output combined with the input via a residual connection.

Multi-head self-attention mechanism in the encoder $TSTAtt(\cdot)$ computes the attention-weighted representations:

$$TSTAtt(\widetilde{\mathbf{T}}_H^i) = \eta_o(Att(\eta_q\widetilde{\mathbf{T}}_H^i, \eta_k\widetilde{\mathbf{T}}_H^i, \eta_v\widetilde{\mathbf{T}}_H^i)), \tag{22}$$

$$Att(\widetilde{\mathbf{T}}_H^i) = softmax\left(\frac{\widetilde{\mathbf{T}}_H^i\widetilde{\mathbf{H}}_L^{i\top}}{\sqrt{d_k}}\right)\widetilde{\mathbf{T}}_H^i, \tag{23}$$

where η_o, η_q, η_k, and η_v are learnable projection matrices, and d_k is the dimensionality of the key vectors.

The final output $\overline{\mathbf{T}}_H \in \mathbb{R}^{N \times D}$ from the $TSTEncoder(\cdot)$ enriched temporal and cross-variable features, which are used for downstream tasks.

D. Privileged Knowledge Distillation

Privileged knowledge distillation emphasizes correlation and feature distillation to transfer the privileged representations from the high-quality teacher to a lightweight student model.

1) Correlation Distillation: Correlation distillation aligns the attention maps between the teacher network's privileged Transformer, $PTEncoder(\cdot)$, and the student's time series Transformer, $TSTEncoder(\cdot)$, leveraging the student model to imitate the teacher's behavior. The teacher's calibrated attention maps guide the student's attention distributions to preserve relational dependencies across features. Specifically, the attention matrices $\mathbf{A}_{\text{PE}} \in \mathbb{R}^{N \times N}$ from $PTEncoder(\cdot)$ and $\mathbf{A}_{\text{TSE}} \in \mathbb{R}^{N \times N}$ from $TSTEncoder(\cdot)$ are averaged across all heads of the last encoder layer to form unified representations. The correlation distillation loss is defined as:

$$\mathcal{L}_{\text{cd}} = \frac{1}{|\mathbf{A}_{\text{PE}}|} \sum_i SL1\left(\mathbf{A}_{\text{PE}}^i - \mathbf{A}_{\text{TSE}}^i\right), \tag{24}$$

where $SL1(\cdot)$, defined in Equation (17), ensures that the student replicates the teacher's contextual understanding of feature features while reducing sensitivity to outliers.

2) Feature Distillation: Feature distillation aligns the embedding spaces of the teacher and student models. The teacher generates privileged embeddings, $\mathbf{E}_{\text{GT}} \in \mathbb{R}^{N \times D}$, from $PTEncoder(\cdot)$, which the student replicates through its time series Transformer, producing embeddings $\overline{\mathbf{T}}_H \in \mathbb{R}^{N \times D}$ from $TSTEncoder(\cdot)$. The feature distillation loss is similarly implemented as a Smooth L1 Loss:

$$\mathcal{L}_{\text{fd}} = \frac{1}{|\mathbf{E}_{\text{GT}}|} \sum_i SL1\left(\mathbf{E}_{\text{GT}}^i - \overline{\mathbf{T}}_H^i\right). \tag{25}$$

Algorithm 2 Privileged Knowledge Distillation
___
Input: Privileged embeddings $\mathbf{E}_{\text{GT}}$, privileged Transformer attentions $\mathbf{A}_{\text{PE}}$, time series embeddings $\overline{\mathbf{T}}_H$, and time series Transformer attentions $\mathbf{A}_{\text{TSE}}$
Output: Distilled student model $DST(\cdot)$
1: **while** not converged **do**
2:    $\mathcal{L}_{\text{cd}} \leftarrow \frac{1}{|\mathbf{A}_{\text{PE}}|} \sum_i SL1(\mathbf{A}_{\text{PE}}^i - \mathbf{A}_{\text{TSE}}^i)$
3:    $\mathcal{L}_{\text{fd}} \leftarrow \frac{1}{|\mathbf{E}_{\text{GT}}|} \sum_i SL1(\mathbf{E}_{\text{GT}}^i - \overline{\mathbf{T}}_H^i)$
4:    $\mathcal{L}_{\text{pkd}} \leftarrow \lambda_c\mathcal{L}_{\text{cd}} + \lambda_f\mathcal{L}_{\text{fd}}$
5:    Update student model $DST(\cdot)$ parameters using $\mathcal{L}_{\text{PKD}}$
6: **end while**
7: **Return:** Distilled student model $DST(\cdot)$

This process ensures that the student effectively captures the enriched features learned by the teacher, enabling robust and accurate knowledge transfer.

The overall distillation loss combines correlation distillation loss $\mathcal{L}_{\text{cd}}$ and feature distillation $\mathcal{L}_{\text{fd}}$ loss to guide the student's learning process. The total knowledge distillation loss is defined as:

$$\mathcal{L}_{\text{PKD}} = \lambda_c\mathcal{L}_{\text{CD}} + \lambda_e\mathcal{L}_{\text{FD}}, \tag{26}$$

where λ_c and λ_e are the hyperparameters that balance the contributions of the correlation distillation loss $\mathcal{L}_{\text{CD}}$ and feature distillation $\mathcal{L}_{\text{FD}}$ loss.

E. Time Series Forecasting

The time series forecasting module leverages the well-learned student model for efficient forecasting.

During the test process, only the student model is employed for inference. Specifically, $\overline{\mathbf{H}}_O$ is the time series embedding from the distilled student model $DST(\cdot)$. Then, the $\overline{\mathbf{H}}_O$ is input into a projection function for future prediction. It can be formulated as follows.

$$\overline{\mathbf{H}}_O = DST(\mathbf{X}_O), \tag{27}$$

$$\hat{\mathbf{X}}_M = \mathbf{W}_t\overline{\mathbf{H}}_O + \mathbf{b}_t, \tag{28}$$

where $\hat{\mathbf{X}}_M \in \mathbb{R}^{M \times N}$ represents the projected forecasts. $\mathbf{W}_t$ and $\mathbf{b}_t$ are the learnable parameters. Finally, the output $\hat{\mathbf{X}}_M$ is normalized. The forecasting loss is SmoothL1 loss:

$$\mathcal{L}_{\text{fcst}} = \frac{1}{M} \sum_{M=1}^{M} SL1(\hat{\mathbf{X}}_M - \mathbf{X}_M,), \tag{29}$$

where M is the number of forecasting sample sizes.

The loss function of the proposed TimeKD consists of four parts: a reconstruction loss $\mathcal{L}_{\text{recon}}$, a correlation distillation loss $\mathcal{L}_{\text{cd}}$, a feature distillation loss $\mathcal{L}_{\text{fd}}$, and a forecasting loss $\mathcal{L}_{\text{fcst}}$. We combine them and obtain the overall loss as follows.

$$\mathcal{L}_{\text{Task}} = \lambda_r\mathcal{L}_{\text{Recon}} + \lambda_p\mathcal{L}_{\text{PKD}} + \lambda_f\mathcal{L}_{\text{Fcst}}, \tag{30}$$

where λ_r, λ_p and λ_f hyperparameters that balance

V. EXPERIMENTAL EVALUATION

A. Experiment Setup

1) Datasets: We conduct experiments with eight widely-used time series datasets for time series forecasting. Specifically, ETTm1, ETTm2, ETTh1, ETTh2 [27], Exchange [61] Weather [62] are utilized for long-term forecasting, while PEMS04, and PEMS08 [29] are employed for short-term forecasting. These datasets span four application domains: electricity, economy, weather, and traffic. Additionally, textual prompts are designed on the time series data.

- **ETT** includes hourly-level datasets (ETTh1 and ETTh2) and 15-minute-level datasets (ETTm1 and ETTm2). Each dataset includes 7 oil and load features of electricity transformers between July 2016 and July 2018.
- **Weather** comprises 21 weather indicators, such as air temperature and humidity, collected in Germany. The data is recorded every 10 minutes.
- **Exchange** consists of daily exchange rates from eight countries, including Australia, the United Kingdom, Canada, Switzerland, China, Japan, New Zealand, and Singapore, covering the period from 1990 to 2016.
- **PEMS** comprises public traffic network data from California, collected in 5-minute intervals. We utiliz two public subsets, PEMS04 and PEMS08, as adopted in existing studies [29], [63].

2) Baselines.: We compare TimeKD with the following time series forecasting methods that include LLM-based methods, i.e., TimeCMA [43], Time-LLM [38], UniTime [49], and OFA [30] and Transformer-based methods, i.e., iTransformer [29], PatchTST [28].

- **TimeCMA [43].** The TimeCMA is an LLM-empowered model that leverages cross-modality alignment to extract effective disentangled features for efficient time series forecasting.
- **Time-LLM [38].** The Time-LLM adapts LLMs for time series forecasting by reprogramming the time series with text prototypes.
- **UniTime [49].** The UniTime employs LLMs to learn from diverse time series datasets, incorporating pure text instructions for cross-domain time series forecasting.
- **OFA [30].** The OFA performs time series forecasting by freezing the attention and feed-forward layers in the LLM while fine-tuning other layers.
- **iTransformer [29].** The iTransformer introduces an inverted embedding to represent long-term dependencies across variables and utilizes the Transformer encoder for time series forecasting.
- **PatchTST [28].** PatchTST introduces a patching mechanism and a channel-independent strategy with Transformer-based models for time series forecasting.

3) Evaluation Metrics: Mean Squared Error (MSE) and Mean Absolute Error (MAE) are adopted as evaluation metrics for forecasting comparison:

$$\text{MSE} = \frac{1}{\mathcal{M}} \sum_{M=1}^{\mathcal{M}} (\hat{\mathbf{X}}_M - \mathbf{X}_M)^2, \qquad (31)$$

$$\text{MAE} = \frac{1}{\mathcal{M}} \sum_{M=1}^{\mathcal{M}} |\hat{\mathbf{X}}_M - \mathbf{X}_M|, \qquad (32)$$

where $\mathcal{M}$ is the number of all predicted values.

4) Implementation Details: To enable fairness, the *test batch size is set to 1* for all methods in the testing phase to avoid the drop last batch trick [64]. All models follow the same experimental setup with an input length of 96 and forecasting horizons of $24, 36, 48, 96$, and 192 across datasets. We train our method using the AdamW optimizer and select the trained model with the lowest average validation loss for testing. We utilize the BERT, GPT-2, and LLaMA-3.2 as large language models. Each experiment is repeated three times with different seeds on NVIDIA A100 GPUs. We set different values to other hyperparameters, e.g., the number of LLM layers, the hidden dimension of the Transformer, and the number of Transformer layers are set to 12, 64, and 2, respectively. Please refer to the associated code repository for details[1].

B. Experiment Results

1) Long-term Forecasting Performance Comparison: We report the MSE and MAE values of the methods in Table I. The best performance by an existing method is underlined, and the overall best performance is marked in bold. The following observations are made.

- TimeKD achieves the best results on all datasets across all forecasting horizons ($FH \in \{24, 36, 48, 96, 192\}$). TimeKD performs better than the best among the baselines up to 9.11% and 7.52% in terms of MSE and MAE, respectively. This is attributed to TimeKD's privileged knowledge distillation, which leverages the calibrated LLMs to extract temporal features while mitigating textual noise in the embeddings. We also observe that the performance improvements on ETTm2 exceed those on other datasets because ETTm2 has a higher sampling frequency and finer-grained numerical records. This verifies TimeKD's ability to learn robust representations for frequently sampled datasets.
- Generally, LLM-based methods, benefiting from the powerful cross-domain knowledge of LLMs, perform better than Transformer-based methods in most cases. It shows the superiority of LLM's generic knowledge and powerful knowledge transfer capabilities. iTransformer performs the worst on all datasets especially on the ETT datasets with fewer variables, as it has a simple model structure without sufficient parameters.
- TimeCMA performs the best among the existing methods due to its prompts that abstract better temporal trends and the design of cross-modality alignment for retrieving robust time series representations.

---

[1]https://github.com/ChenxiLiu-HNU/TimeKD

TABLE I: Long-term Forecasting performance comparisons. The input length is 96 for all datasets.

Datasets		TimeKD MSE	TimeKD MAE	TimeCMA MSE	TimeCMA MAE	TimeLLM MSE	TimeLLM MAE	UniTime MSE	UniTime MAE	OFA MSE	OFA MAE	iTransformer MSE	iTransformer MAE	PatchTST MSE	PatchTST MAE
ETTm1	24	**0.202**	**0.258**	0.208	0.281	0.263	0.232	0.228	0.301	0.221	0.293	0.223	0.296	0.210	0.283
	36	**0.259**	**0.302**	0.267	0.319	0.296	0.344	0.273	0.330	0.269	0.324	0.274	0.331	0.272	0.325
	48	**0.272**	**0.323**	0.282	0.329	0.295	0.343	0.298	0.346	0.292	0.338	0.312	0.356	0.286	0.335
	96	**0.305**	**0.342**	0.312	0.351	0.359	0.381	0.328	0.367	0.326	0.362	0.334	0.373	0.320	0.359
	192	**0.355**	**0.367**	0.361	0.378	0.383	0.393	0.368	0.387	0.367	0.382	0.377	0.391	0.362	0.381
	Avg	**0.279**	**0.318**	0.286	0.331	0.319	0.357	0.299	0.346	0.295	0.340	0.304	0.349	0.290	0.337
ETTm2	24	**0.083**	**0.184**	0.092	0.190	0.102	0.198	0.105	0.201	0.103	0.201	0.104	0.200	0.099	0.193
	36	**0.109**	**0.204**	0.116	0.211	0.122	0.218	0.124	0.221	0.124	0.222	0.123	0.219	0.119	0.215
	48	**0.125**	**0.214**	0.130	0.225	0.137	0.230	0.140	0.234	0.138	0.236	0.139	0.234	0.134	0.229
	96	**0.170**	**0.251**	0.173	0.258	0.193	0.280	0.181	0.263	0.176	0.261	0.180	0.264	0.177	0.260
	192	**0.225**	**0.296**	0.238	0.301	0.257	0.318	0.248	0.308	0.243	0.305	0.250	0.309	0.246	0.305
	Avg	**0.142**	**0.230**	0.150	0.237	0.162	0.249	0.160	0.245	0.157	0.245	0.159	0.245	0.155	0.240
ETTh1	24	**0.290**	**0.346**	0.296	0.352	0.325	0.370	0.359	0.390	0.295	0.351	0.304	0.359	0.321	0.365
	36	**0.314**	**0.360**	0.318	0.363	0.348	0.383	0.380	0.401	0.318	0.364	0.328	0.372	0.343	0.377
	48	**0.325**	**0.368**	0.340	0.375	0.358	0.389	0.392	0.409	0.330	0.371	0.342	0.379	0.354	0.384
	96	**0.364**	**0.392**	0.373	**0.391**	0.398	0.440	0.427	0.430	0.374	0.394	0.386	0.405	0.395	0.407
	192	**0.421**	0.422	0.427	**0.421**	0.451	0.440	0.465	0.452	0.429	0.429	0.441	0.436	0.445	0.434
	Avg	**0.343**	**0.378**	0.351	0.380	0.376	0.398	0.405	0.416	0.349	0.382	0.360	0.390	0.372	0.393
ETTh2	24	**0.167**	**0.259**	0.171	0.262	0.182	0.272	0.178	0.269	0.172	0.263	0.186	0.276	0.173	0.265
	36	**0.200**	**0.272**	0.203	0.281	0.211	0.292	0.210	0.290	0.206	0.287	0.217	0.297	0.204	0.285
	48	**0.220**	**0.294**	0.225	0.298	0.232	0.302	0.235	0.306	0.230	0.305	0.242	0.313	0.226	0.300
	96	**0.278**	**0.332**	0.286	0.336	0.295	0.346	0.300	0.348	0.301	0.352	0.297	0.349	0.291	0.341
	192	**0.356**	**0.381**	0.363	0.387	0.386	0.399	0.374	0.398	0.386	0.406	0.380	0.400	0.377	0.393
	Avg	**0.245**	**0.308**	0.250	0.313	0.261	0.322	0.259	0.322	0.259	0.323	0.264	0.327	0.254	0.317
Weather	24	**0.101**	**0.131**	0.105	0.136	0.106	0.140	0.107	0.142	0.110	0.143	0.107	0.140	0.105	0.137
	36	**0.119**	**0.150**	0.122	0.156	0.124	0.161	0.126	0.166	0.134	0.172	0.125	0.159	0.126	0.163
	48	**0.130**	**0.166**	0.135	0.171	0.142	0.154	0.140	0.182	0.148	0.189	0.136	0.173	0.141	0.181
	96	**0.163**	**0.207**	0.167	0.211	0.195	0.233	0.174	0.220	0.183	0.223	0.174	0.214	0.177	0.218
	192	**0.208**	**0.247**	0.212	0.253	0.240	0.269	0.222	0.260	0.229	0.261	0.221	0.254	0.223	0.258
	Avg	**0.144**	**0.180**	0.148	0.185	0.161	0.191	0.154	0.194	0.161	0.198	0.153	0.188	0.154	0.191
Exchange	24	**0.022**	**0.104**	0.026	0.109	0.028	0.119	0.029	0.119	0.025	0.108	0.026	0.110	0.025	0.108
	36	**0.031**	**0.124**	0.034	0.128	0.039	0.138	0.038	0.137	0.033	0.127	0.036	0.133	0.034	0.129
	48	**0.041**	**0.142**	0.044	0.147	0.050	0.158	0.048	0.154	0.045	0.148	0.045	0.148	0.045	0.148
	96	**0.082**	**0.202**	0.086	0.208	0.090	0.210	0.091	0.211	0.084	0.206	0.086	0.206	0.085	0.207
	192	**0.173**	**0.296**	0.177	0.299	0.181	0.302	0.182	0.304	0.179	0.300	0.177	0.300	0.181	0.301
	Avg	**0.070**	**0.174**	0.073	0.178	0.078	0.185	0.078	0.185	0.073	0.177	0.074	0.179	0.074	0.179

TABLE II: Short-term forecasting performance comparison with input length of 96 and forecasting horizon of 12.

Datasets	TimeKD MSE	TimeKD MAE	TimeCMA MSE	TimeCMA MAE	TimeLLM MSE	TimeLLM MAE	UniTime MSE	UniTime MAE	OFA MSE	OFA MAE	iTransformer MSE	iTransformer MAE	PatchTST MSE	PatchTST MAE
PEMS04	**0.066**	**0.165**	0.074	0.178	0.083	0.187	0.087	0.193	0.089	0.195	0.078	0.183	0.150	0.224
PEMS08	**0.063**	**0.161**	0.076	0.180	0.085	0.189	0.089	0.196	0.092	0.199	0.079	0.182	0.168	0.232

2) Short-term Forecasting Performance Comparison: As shown in Table II, TimeKD achieves the best performance among all baselines on the PEMS04 and PEMS08 datasets. Specifically, TimeKD outperforms TimeCMA with 10.81% and 10.26% reductions in MSE on PEMS04 and PEMS08, respectively. The third-best model, iTransformer, is surpassed by TimeKD with 15.38% and 11.39% improvements on PEMS04 and PEMS08, respectively. The superior results of the top three models (TimeKD, TimeCMA, and iTransformer) can be attributed to their use of inverted embeddings, which effectively capture the spatial dependencies among traffic sensors. In contrast, Time-LLM, UniTime, OFA, and PatchTST handle each sensor independently, without considering spatial interactions, and thus achieve relatively lower performance.

3) Ablation Studies of Model Design: To gain insight into the effects of the different components of TimeKD, including privileged information, calibrated attention, language model, subtractive cross attention, correlation distillation, and feature distillation. We evaluate the following variants.

- **w/o_PI**. TimeKD without privileged information (e.g., ground truth prompts), where only feed historical data into the teacher model. This variant is the traditional
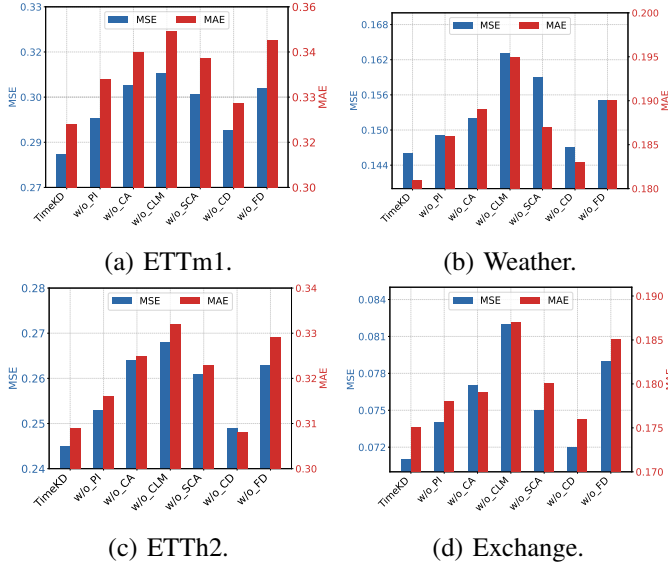
(a) ETTm1.   (b) Weather.

(c) ETTh2.   (d) Exchange.

Fig. 6: Performance of TimeKD and its variants on Four Datasets.

TABLE III: Ablation study of LLMs within TimeKD on Exchange. The forecasting horizon is 24.

	LLMs		Exchange	
Models	Model sizes (B)		MSE	MAE
BERT	0.110		0.032	0.125
GPT-2	0.117		<u>0.024</u>	<u>0.105</u>
LLaMA-3.2	3		**0.020**	**0.102**

TABLE IV: Efficiency analysis of TimeKD and baselines. The forecasting horizon is 96

Models	ETTm1			
	Trainabl. Param.	Train. Time	Mem.	Infer. Speed
iTransformer	**0.22**	**12.97**	1,722	<u>0.08</u>
Time-LLM	44.66	4799.64	28,882	1.08
UniTime	108.54	2472.15	4,168	0.39
OFA	1.75	425.12	910	0.18
TimeCMA	17.99	50.13	<u>821</u>	0.09
TimeKD	<u>1.72</u>	<u>49.78</u>	**730**	**0.06**

teacher in Figure 1.

- **w/o_CA**. TimeKD without calibrated attention, using the original multi-head attention mechanism.
- **w/o_CLM**. TimeKD without calibrated language models, where the teacher model does not leverage LLMs for textual prompt encoding.
- **w/o_SCA**. TimeKD without subtractive cross attention, using the direct subtraction of embeddings replaces the subtractive cross attention.
- **w/o_CD**. TimeKD without correlation distillation, eliminating direct interactions between the privileged transformer and the time series transformer.
- **w/o_FD**. TimeKD without the feature distillation, removing alignments between the outputs of the privileged transformer and the time series transformer.

Figure 6 shows the average results on ETTm1, ETTh2, Weather, and Exchange datasets across all forecasting horizons. Regardless of the datasets, TimeKD outperforms its counterparts without CD and PI. This shows that correlation distillation and privileged information are useful for effective time series forecasting. Notably, when the teacher model only includes historical data without the ground truth prompt, the performance of TimeKD is reduced, which indicates the significance of the privileged information during the distillation. In the ETTm1, TimeKD achieves MSE and MAE reductions by up to 8.2% and 6.5%, respectively, compared with w/o_SCA. In addition, TimeKD performs better than w/o_CA by at least 8.9% and 8.4% in terms of MSE and MAE, respectively, indicating the effectiveness of the subtractive cross attention and calibrated attention. Further, w/o_FD underperforms compared to most other variants, while w/o_CLM demonstrates the weakest performance among all, emphasizing the contribution of the calibrated language model.

4) Ablation Studies of Open-Source LLMs: Table III presents an ablation study on LLM selection in TimeKD, comparing three open-source models. BERT serves as a smaller baseline, GPT-2 has slightly more parameters and better prediction results. LLaMA-3.2 achieves the best performance. This trend suggests that larger LLMs capture more complex linguistic patterns, enhancing distillation. However, LLaMA-3.2 provides marginal accuracy gains at a significantly higher computational cost. With a hidden dimension of 4096 versus GPT-2's 768, it requires more memory and longer inference times. Given GPT-2's simplicity and strong performance, we adopt it as our TimeKD backbone for better efficiency.

5) Resource Efficiency: We assess the resource efficiency of TimeKD and the baselines on the ETTm1 across four key metrics: Trainable Parameters (Trainabl. Param.) in millions, Training Time (Train. Time) of an epoch in seconds, Memory Usage (Mem.) in MiB, and Inference Speed (Infer. Speed) in seconds per iteration, all evaluated on NVIDIA A100 GPUs. To ensure a fair comparison of memory usage, we set the training batch size to 8 for all baselines, ensuring that each iteration processes exactly 8 samples.

The efficiency comparison results are presented in Table IV: *(1) TimeKD achieves the lowest memory consumption and the highest inference speed among all methods*, including smaller models like iTransformer. This is due to the design of the last token extractor in the teacher model and the simplified hyperparameter settings in the student model, which has already benefited from learning pre-trained knowledge from the LLM. *(2) TimeKD has the lowest trainable parameter count and training time among the LLM-based methods*, and is the second only to iTransformer. This is due to the additional learnable parameters introduced by the privileged Transformer in the teacher model and the privileged knowledge distillation, which result in slightly higher training costs compared to the standalone student model.
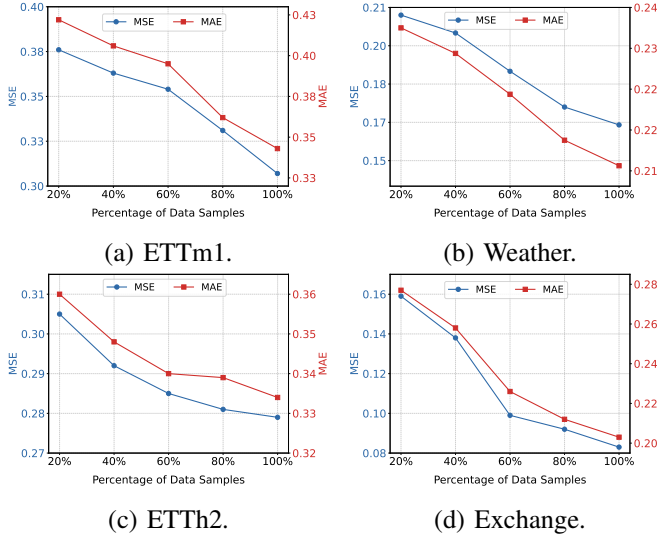
(a) ETTm1. (b) Weather.

(c) ETTh2. (d) Exchange.

Fig. 7: Effect of Different Training Data on Four Datasets.

6) Scalability: The scalability analysis of TimeKD on ETTm1, ETTh2, Weather, and Exchange datasets with a forecasting horizon set to 96 demonstrates its robustness under different data scarcity scenarios, as shown in Figure 7. As the percentage of available data samples increases from 20% to 100%, MSE and MAE consistently decrease for both datasets. This indicates that TimeKD effectively leverages the additional data to improve forecasting accuracy, showing adaptability to varying data scales. Notably, the performance improvement is more pronounced in the ETTm1, reflecting its ability to capture complex temporal patterns in frequently sampled data. Similarly, for the Weather, the steady decline in the metrics highlights the capability of TimeKD to generalize across diverse datasets. These results confirm the scalability of TimeKD under limited data conditions, making it suitable for real-world applications with varying data availability.

7) Few-shot forecasting: To demonstrate whether the proposed TimeKD has remarkable few-shot learning capabilities, we evaluate TimeKD and existing methods with limited training data, i.e., the first 10% of the training data, and set the forecasting horizon to 96. The prediction results are given in Table V. Overall, TimeKD consistently surpasses other competitive baselines, indicating its stable and superior performance, especially on ETTm1. Specifically, TimeKD performs better than TimeCMA by up to 3.18% and 9.79% in terms of MSE and MAE, respectively, illustrating the importance of effective knowledge distillation in enhancing model performance under data scarcity. The LLM-based method TimeCMA achieves the best performance among existing methods (except MAE on the ETTh2). Furthermore, LLM-based methods perform better than Transformer-based methods in most scenarios. The reason is that these LLM-based methods leverage the pre-trained knowledge of large language models, enabling them to capture temporal patterns even with limited training data.
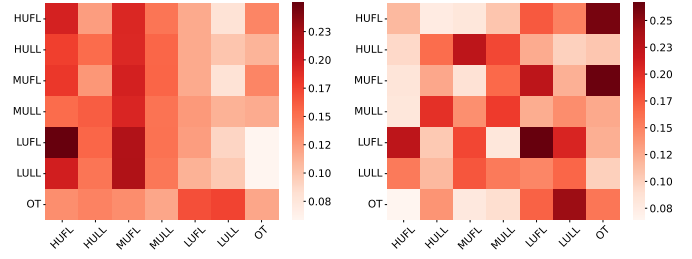

(a) Privileged Transformer. (b) Time Series Transformer.

Fig. 8: Attention maps from Transformers on ETTm1, capturing the pairwise relations of variables.

8) Zero-shot forecasting: Beyond few-shot forecasting, LLMs hold potential as effective zero-shot reasoners. We assess the zero-shot capabilities of TimeKD with existing methods. The MSE and MAE values are given in Table VI. We observe that TimeKD outperforms existing methods consistently, showing its powerful zero-shot learning capabilities. Specifically, TimeKD performs better than TimeCMA by up to 9.15% and 11.4% in terms of MSE and MAE, respectively. The reason is that TimeKD effectively transfers knowledge across datasets through privileged knowledge distillation, enabling it to generalize better in zero-shot scenarios. By extracting common temporal patterns between different datasets, TimeKD enhances its ability to forecast accurately without training directly on the target dataset.

9) Attention Maps Visualization: We visualize attention maps from the privileged Transformer and the time series Transformer on the ETTm1 with a forecasting horizon of 96. In Figures 8 (a) and (b), each row corresponds to a variable and its attention to other variables. The privileged attention is learned from the LLM, while time series attention is extracted from a Pre-LN Transformer encoder. The visualizations highlight distinct behaviors in capturing multivariable dependencies. The LLM textual attention is universal and captures global dependencies between variables due to the pre-trained knowledge from LLMs and privileged information from the input prompts. The attention map from the time series Transformer is local and variable-specific. By distilling the correlations from the privileged Transformer into the time series Transformer, TimeKD effectively leverages local and global dependencies and enhances forecasting accuracy.

10) Feature Visualization: We visualize feature outputs from the privileged Transformer and time series Transformer on ETTm1 with a forecasting horizon of 96. We multiply the feature matrix by its transpose to compute pairwise interactions between variables, producing a self-relation feature matrix, as shown in Figure 9. In Figure 9 (a), the features are from the privileged Transformer, and show a more comprehensive and balanced pattern of interactions across variables, benefiting from the global contextual knowledge encoded in the LLMs. In contrast, Figure 9 (b), which visualizes features from the time series Transformer, reveals a localized focus with sparser interactions, reflecting the model's emphasis on

TABLE V: Few-shot forecasting on 10% training data. The input length and forecasting horizon are set to 96 time steps.

Datasets	TimeKD		TimeCMA		Time-LLM		UniTime		OFA		iTransformer		PatchTST	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.429**	**0.396**	<u>0.442</u>	<u>0.439</u>	0.587	0.491	0.559	0.486	0.615	0.497	0.565	0.484	0.558	0.478
ETTm2	**0.183**	**0.261**	<u>0.185</u>	<u>0.265</u>	0.189	0.270	0.186	0.267	0.187	0.266	0.194	0.277	0.189	0.268
ETTh1	**0.421**	**0.415**	<u>0.431</u>	<u>0.422</u>	0.498	0.462	0.502	0.467	0.462	0.449	0.537	0.493	0.433	0.428
ETTh2	**0.304**	**0.341**	<u>0.314</u>	0.355	0.329	0.367	0.331	0.368	0.327	0.359	0.341	0.378	<u>0.314</u>	<u>0.354</u>

TABLE VI: Zero-shot Forecasting Results on ETT. The notation "trained dataset" → "test dataset" indicates the dataset used for training and the one used for testing, respectively. The forecasting horizon is set to 96 time steps.

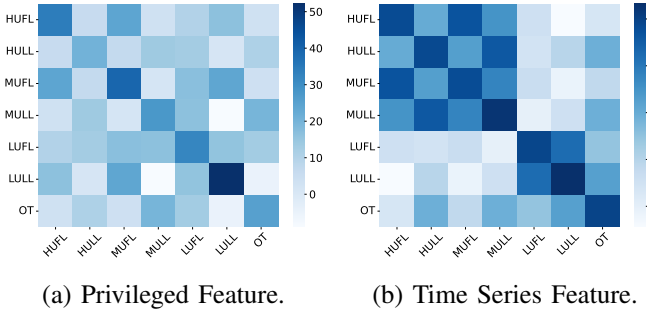| Model	TimeKD		TimeCMA		Time-LLM		UniTime		OFA		iTransformer		PatchTST	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1 → ETTm2	**0.189**	**0.263**	<u>0.192</u>	<u>0.266</u>	0.194	0.267	0.197	0.273	0.196	0.267	0.202	0.279	0.193	0.269
ETTm2 → ETTm1	**0.487**	**0.439**	<u>0.495</u>	<u>0.443</u>	0.514	0.441	0.529	0.449	0.519	0.457	0.679	0.52	0.554	0.460
ETTh1 → ETTh2	**0.268**	**0.336**	<u>0.295</u>	<u>0.341</u>	0.300	0.346	0.308	0.349	0.297	0.344	0.296	0.344	0.296	0.343
ETTh2 → ETTh1	**0.375**	**0.391**	<u>0.389</u>	<u>0.440</u>	0.522	0.479	0.537	0.490	0.512	0.472	0.575	0.515	0.485	0.461

(a) Privileged Feature.   (b) Time Series Feature.

Fig. 9: Features from the privileged Transformer and the time series Transformer on ETTm1.

(a) HUFL.   (b) MUFL.

(c) LUFL.   (d) OT.

Fig. 10: Ground truth vs. prediction visualization on ETTh1.

local temporal dependencies.

11) Ground Truth vs. Prediction: To evaluate whether the TimeKD effectively captures temporal patterns, we visualize the predicted time series and the ground truth on ETTh1, as shown in Figure 10. The four subfigures represent the results for the four variables: high useful load (HUFL), middle useful load (MUFL), low useful load (LUFL), and oil temperature (OT), respectively. From these visualizations, it is evident that the predicted blue curves closely follow the ground truth orange curves, particularly in capturing periodic trends and fluctuations over time. This alignment demonstrates that TimeKD successfully learns and preserves key temporal dependencies, further validating its ability to produce accurate forecasts across multiple time series variables.

VI. CONCLUSION

This paper introduces TimeKD, a novel multivariate time series (MTSF) forecasting framework that integrates calibrated language models with privileged knowledge distillation. TimeKD is designed with two key components: a cross-modality teacher model and a lightweight student model. In the cross-modality teacher model, we propose a calibrated la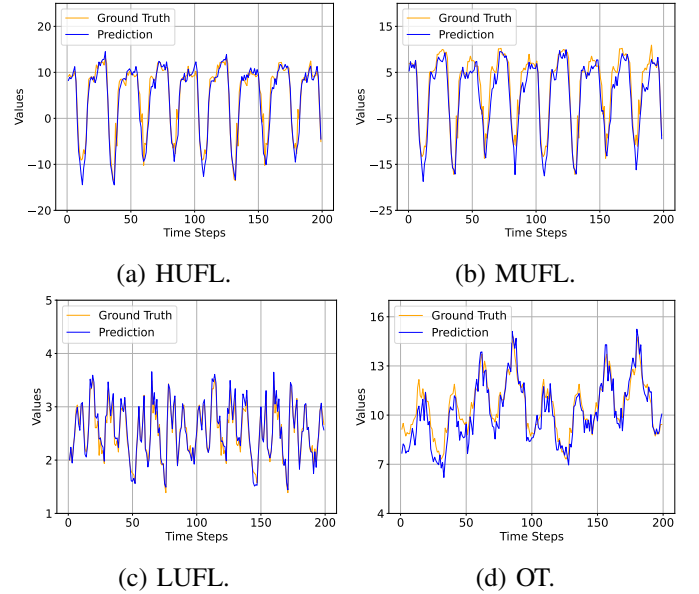nguage model and subtractive cross-attention. The calibrated language model extracts robust future representations based on the pre-trained knowledge of LLMs and the privileged textual prompts. The subtractive cross attention is proposed to purify the representations to align the time series data. We propose an innovative privileged knowledge distillation including correlation and feature distillation, that transfers the representations from the teacher model to the lightweight student model. We for the first time apply privileged distillation calibration to open-source LLMs for MTSF. Extensive experiments on real datasets from diverse domains demonstrate the effectiveness and efficiency of TimeKD.

VII. ACKNOWLEDGMENT

References

[1] D. Campos, M. Zhang, B. Yang, T. Kieu, C. Guo, and C. S. Jensen, "Lightts: Lightweight time series classification with adaptive ensemble distillation," *SIGMOD*, vol. 1, no. 2, pp. 1–27, 2023.

[2] C. Liu, Z. Xiao, C. Long, D. Wang, T. Li, and H. Jiang, "MVCAR: Multi-view collaborative graph network for private car carbon emission prediction," *TITS*, 2024.

[3] H. Miao, Z. Liu, Y. Zhao, C. Guo, B. Yang, K. Zheng, and C. S. Jensen, "Less is more: Efficient time series dataset condensation via two-fold modal matching," *PVLDB*, vol. 18, no. 2, pp. 226–238, 2024.

[4] X. Wang, H. Zhang, P. Wang, Y. Zhang, B. Wang, Z. Zhou, and Y. Wang, "An observed value consistent diffusion model for imputing missing values in multivariate time series," in *SIGKDD*, 2023, pp. 2409–2418.

[5] L. Tran, M. Y. Mun, M. Lim, J. Yamato, N. Huh, and C. Shahabi, "Deeptrans: a deep learning system for public bus travel time estimation using traffic forecasting," *PVLDB*, vol. 13, no. 12, pp. 2957–2960, 2020.

[6] H. Yuan, G. Li, Z. Bao, and L. Feng, "Effective travel time estimation: When historical trajectories over road networks matter," in *SIGMOD*, 2020, pp. 2135–2149.

[7] C. Liu, J. Cai, D. Wang, J. Tang, L. Wang, H. Chen, and Z. Xiao, "Understanding the regular travel behavior of private vehicles: An empirical evaluation and a semi-supervised model," *IEEE Sensors Journal*, vol. 21, no. 17, pp. 19 078–19 090, 2021.

[8] C. Liu, Z. Xiao, D. Wang, M. Cheng, H. Chen, and J. Cai, "Foreseeing private car transfer between urban regions with multiple graph-based generative adversarial networks," *World Wide Web*, vol. 25, no. 6, pp. 2515–2534, 2022.

[9] Y. Fang, Y. Liang, B. Hui, Z. Shao, L. Deng, X. Liu, X. Jiang, and K. Zheng, "Efficient large-scale traffic forecasting with transformers: A spatial data management perspective," in *KDD*, 2025.

[10] C. Liu, Z. Xiao, D. Wang, L. Wang, H. Jiang, H. Chen, and J. Yu, "Exploiting spatiotemporal correlations of arrive-stay-leave behaviors for private car flow prediction," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 2, pp. 834–847, 2022.

[11] Y. Zhang, X. Wang, X. Yu, Z. Sun, K. Wang, and Y. Wang, "Drawing informative gradients from sources: A one-stage transfer learning framework for cross-city spatiotemporal forecasting," in *AAAI*, 2025.

[12] Y. Zhang, X. Wang, P. Wang, B. Wang, Z. Zhou, and Y. Wang, "Modeling spatio-temporal mobility across data silos via personalized federated learning," *TMC*, vol. 23, no. 12, pp. 15 289–15 306, 2024.

[13] Z. Xu, L. Li, M. Zhang, Y. Xu, and X. Zhou, "Managing the future: Route planning influence evaluation in transportation systems," in *ICDE*, 2024, pp. 4558–4572.

[14] J. Cai, D. Wang, H. Chen, C. Liu, and Z. Xiao, "Modeling dynamic spatiotemporal user preference for location prediction: a mutually enhanced method," *World Wide Web*, vol. 27, no. 2, p. 14, 2024.

[15] H. Chen, D. Wang, and C. Liu, "Towards semantic travel behavior prediction for private car users," in *HPCC*, 2020, pp. 950–957.

[16] C. Chang, C.-T. Chan, W.-Y. Wang, W.-C. Peng, and T.-F. Chen, "TimeDRL: Disentangled representation learning for multivariate time-series," in *ICDE*, 2024, pp. 625–638.

[17] R. Zha, L. Zhang, S. Li, J. Zhou, T. Xu, H. Xiong, and E. Chen, "Scaling up multivariate time series pre-training with decoupled spatial-temporal representations," in *ICDE*, 2024, pp. 667–678.

[18] Z. Shao, F. Wang, Y. Xu, W. Wei, C. Yu, Z. Zhang, D. Yao, T. Sun, G. Jin, X. Cao *et al.*, "Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis," *TKDE*, 2024.

[19] Y. Sakurai, Y. Matsubara, and C. Faloutsos, "Mining and forecasting of big time-series data," in *SIGMOD*, 2015, pp. 919–922.

[20] R. Xu, H. Miao, S. Wang, P. S. Yu, and J. Wang, "PeFAD: A parameter-efficient federated framework for time series anomaly detection," in *SIGKDD*, 2024, pp. 3621–3632.

[21] C. Liu, Z. Xiao, W. Long, T. Li, H. Jiang, and K. Li, "Vehicle trajectory data processing, analytics, and applications: A survey," *CSUR*, 2025.

[22] J. Xiao, Z. Xiao, D. Wang, V. Havyarimana, C. Liu, C. Zou, and D. Wu, "Vehicle trajectory interpolation based on ensemble transfer regression," *TITS*, vol. 23, no. 7, pp. 7680–7691, 2022.

[23] Z. Liu, H. Miao, Y. Zhao, C. Liu, K. Zheng, and H. Li, "LightTR: A lightweight framework for federated trajectory recovery," in *ICDE*, 2024.

[24] S. Yang, Q. Su, Z. Li, Z. Li, H. Mao, C. Liu, and R. Zhao, "Sql-to-schema enhances schema linking in text-to-sql," in *DEXA*, vol. 14910, 2024, pp. 139–145.

[25] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen, "Foundation models for time series analysis: A tutorial and survey," in *SIGKDD*, 2024, pp. 6555–6565.

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

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

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

[29] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "iTransformer: Inverted transformers are effective for time series forecasting," in *ICLR*, 2023.

[30] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One Fits All: Power general time series analysis by pretrained LM," in *NeurIPS*, 2023.

[31] C. Chang, W.-C. Peng, and T.-F. Chen, "LLM4TS: Two-stage fine-tuning for time-series forecasting with pre-trained llms," *arXiv*, 2023.

[32] C. Sun, Y. Li, H. Li, and S. Hong, "TEST: Text prototype aligned embedding to activate llm's ability for time series," in *ICLR*, 2023.

[33] H. Xue and F. D. Salim, "PromptCast: A new prompt-based learning paradigm for time series forecasting," *TKDE*, pp. 1–14, 2023.

[34] H. Miao, Y. Zhao, C. Guo, B. Yang, K. Zheng, and C. S. Jensen, "Spatiotemporal prediction on streaming data: A unified federated continuous learning framework," *TKDE*, 2025.

[35] I. Trummer, "Large language models: Principles and practice," in *ICDE*, 2024, pp. 5354–5357.

[36] M. Zhang, Z. Ji, Z. Luo, Y. Wu, and C. Chai, "Applications and challenges for large language models: From data management perspective," in *ICDE*, 2024, pp. 5530–5541.

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

[38] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan *et al.*, "Time-LLM: Time series forecasting by reprogramming large language models," in *ICLR*, 2024.

[39] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *IJCV*, vol. 129, no. 6, pp. 1789–1819, 2021.

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

[41] W. Lu, J. Wang, X. Sun, Y. Chen, X. Ji, Q. Yang, and X. Xie, "Diversify: A general framework for time series out-of-distribution detection and generalization," *TPAMI*, 2024.

[42] C. Liu, S. He, Q. Zhou, S. Li, and W. Meng, "Large language model guided knowledge distillation for time series anomaly detection," in *IJCAI*, 2024, pp. 2162–2170.

[43] C. Liu, Q. Xu, H. Miao, S. Yang, L. Zhang, C. Long, Z. Li, and R. Zhao, "TimeCMA: Towards llm-empowered multivariate time series forecasting via cross-modality alignment," 2025.

[44] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, "Unifying distillation and privileged information," in *ICLR*, 2016.

[45] R.-G. Cirstea, T. Kieu, C. Guo, B. Yang, and S. J. Pan, "Enhancenet: Plugin neural networks for enhancing correlated time series forecasting," in *ICDE*, 2021, pp. 1739–1750.

[46] G. Jin, C. Liu, Z. Xi, H. Sha, Y. Liu, and J. Huang, "Adaptive dual-view wavenet for urban spatial-temporal event prediction," *Inf. Sci.*, vol. 588, pp. 315–330, 2022.

[47] C. Liu, D. Wang, H. Chen, and R. Li, "Study of forecasting urban private car volumes based on multi-source heterogeneous data fusion." *Journal on Communication*, vol. 42, no. 3, 2021.

[48] D. Cao, F. Jia, S. O. Arik, T. Pfister, Y. Zheng, W. Ye, and Y. Liu, "TEMPO: Prompt-based generative pre-trained transformer for time series forecasting," in *ICLR*, 2023.

[49] X. Liu, J. Hu, Y. Li, S. Diao, Y. Liang, B. Hooi, and R. Zimmermann, "Unitime: A language-empowered unified model for cross-domain time series forecasting," in *WWW*, 2024, pp. 4095–4106.

[50] Z. Pan, Y. Jiang, S. Garg, A. Schneider, Y. Nevmyvaka, and D. Song, "S^2IP-LLM: Semantic space informed prompt learning with llm for time series forecasting," in *ICLR*, 2024.

[51] Y. Liu, G. Qin, X. Huang, J. Wang, and M. Long, "AutoTimes: Autoregressive time series forecasters via large language models," in *NeurIPS*, 2024.

[52] C. Liu, S. Yang, Q. Xu, Z. Li, C. Long, Z. Li, and R. Zhao, "Spatial-temporal large language model for traffic prediction," in *MDM*, 2024.

[53] Y. Gu, L. Dong, F. Wei, and M. Huang, "MiniLLM: Knowledge distillation of large language models," in *ICLR*, 2024.

[54] X. Li, L. Lin, S. Wang, and C. Qian, "Self-improving teacher cultivates better student: Distillation calibration for multimodal large language models," in *SIGIR*, 2024, pp. 882–892.

[55] M. Kang, S. Lee, J. Baek, K. Kawaguchi, and S. J. Hwang, "Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks," *NeurIPS*, vol. 36, 2024.

[56] J. Zhao, W. Zhao, A. Drozdov, B. Rozonoyer, M. A. Sultan, J. Y. Lee, M. Iyyer, and A. McCallum, "Multistage collaborative knowledge distillation from a large language model for semi-supervised sequence generation," in *ACL*, 2024, pp. 14 201–14 214.

[57] B. Pan, Z. Zhang, Y. Zhang, Y. Hu, and L. Zhao, "Distilling large language models for text-attributed graph learning," in *CIKM*, 2024, pp. 1836–1845.

[58] L. Li, Y. Zhang, and L. Chen, "Prompt distillation for efficient llm-based recommendation," in *CIKM*, 2023, pp. 1348–1357.

[59] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *ICML*, vol. 119, 2020, pp. 10 524–10 533.

[60] T. Kim, J. Kim, Y. Tae, C. Park, J. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *ICLR*, 2022.

[61] G. Lai, W. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *SIGIR*, 2018, pp. 95–104.

[62] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *AAAI*, 2021, pp. 22 419–22 430.

[63] Y. Hu, G. Zhang, P. Liu, D. Lan, N. Li, D. Cheng, T. Dai, S.-T. Xia, and S. Pan, "TimeFilter: Patch-specific spatial-temporal graph filtration for time series forecasting," *arXiv*, 2025.

[64] X. Qiu, J. Hu, L. Zhou, X. Wu, J. Du, B. Zhang, C. Guo, A. Zhou, C. S. Jensen, Z. Sheng, and B. Yang, "TFB: towards comprehensive and fair benchmarking of time series forecasting methods," in *PVLDB*, 2024.