
CALF: Aligning LLMs for Time Series Forecasting via Cross-modal Fine-Tuning

Peiyuan Liu^{1,*} Hang Guo^{1,*} Tao Dai^{2,✉} Naiqi Li^{1,✉} Jigang Bao¹
Xudong Ren¹ Yong Jiang¹ Shu-tao Xia¹

¹Tsinghua Shenzhen International Graduate School ²Shenzhen University
{peiyuanliu.edu, cshguo, daitao.edu, linaiqi.thu}@gmail.com
{baojg19, rxd21}@mails.tsinghua.edu.cn
{jiangy, xiast}@sz.tsinghua.edu.cn

Abstract

Deep learning (e.g., Transformer) has been widely and successfully used in multivariate time series forecasting (MTSF). Unlike existing methods that focus on training models from a single modal of time series input, large language models (LLMs) based MTSF methods with cross-modal text and time series input have recently shown great superiority, especially with limited temporal data. However, current LLM-based MTSF methods usually focus on adapting and fine-tuning LLMs, while neglecting the *distribution discrepancy* between textual and temporal input tokens, thus leading to sub-optimal performance. To address this issue, we propose a novel **Cross-ModAl LLM Fine-Tuning (CALF)** framework for MTSF by reducing the distribution discrepancy between textual and temporal data, which mainly consists of the temporal target branch with temporal input and the textual source branch with aligned textual input. To reduce the distribution discrepancy, we develop the cross-modal match module to first align cross-modal input distributions. Additionally, to minimize the modality distribution gap in both feature and output spaces, feature regularization loss is developed to align the intermediate features between the two branches for better weight updates, while output consistency loss is introduced to allow the output representations of both branches to correspond effectively. Thanks to the modality alignment, CALF establishes state-of-the-art performance for both long-term and short-term forecasting tasks with low computational complexity, and exhibiting favorable few-shot and zero-shot abilities similar to that in LLMs. Code is available at <https://github.com/Hank0626/CALF>.

1 Introduction

Multivariate time series forecasting (MTSF) plays a crucial role in the domain of time series analysis and has further boasted a wide range of real-world applications including weather forecasting [1], energy prediction [2], financial modeling [3]. To achieve more accurate forecasting performance, numerous deep learning-based MTSF methods trained on a single modal of time series input have been developed in recent years [4, 5, 6, 7, 8, 9, 10, 11] and have gained great success.

However, previous single-modal MTSF methods [12] may suffer from overfitting problems, due to the limited training data, thus limiting their real applications. To relieve such issues, some pioneering works attempt to introduce the powerful Large Language Models (LLMs) models in time series forecasting by employing the strong context modeling ability of LLMs. For example, Zhou et al. [13] proposed a unified time series analysis framework by adapting and fine-tuning LLMs. Building upon

*Equal Contribution

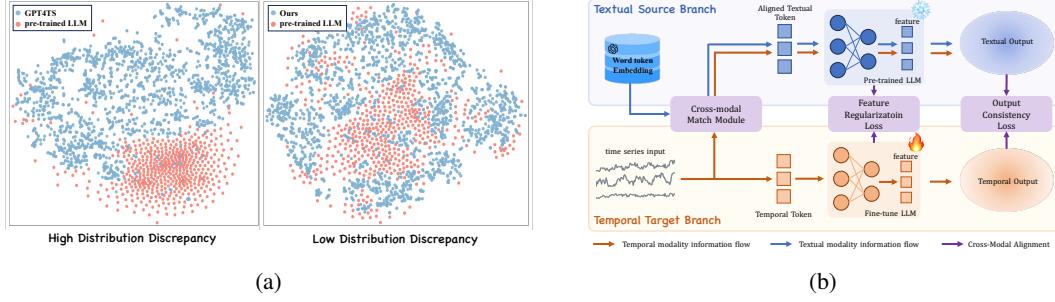


Figure 1: (a) The t-SNE visualization of pre-trained word token embeddings of LLM with temporal tokens of ETTh2 dataset from GPT4TS [13] (Left) and our method (Right). Our method shows more cohesive integration, indicating effective modality alignment. Appendix A shows more results. (b) Conceptual illustration of cross-modal fine-tuning technique.

this, other works have introduced additional enhancements to further expand the capabilities of LLMs in time series forecasting, including refining fine-tuning methods [14], sequence decomposition [15], and the incorporation of textual prompts [12]. Benefiting from the large-scale pre-training, LLM-based methods not only exhibit strong context modeling capabilities but also help mitigate the problem of overfitting.

Despite the great success of LLM-based MTSF methods, existing LLM-based MTSF methods usually focus on adapting and fine-tuning LLMs, while neglecting the *distribution discrepancy* between textual and temporal input tokens, thus leading to sub-optimal performance. In practice, current LLM-based methods typically treat pre-trained LLMs as well-initialized forecasting models and project time series data using a simple linear layer as input for the LLMs. While this straightforward approach is intuitive, it can lead to sub-optimal results due to significant distribution discrepancies between textual and temporal data. As shown in Fig. 1a, we show the distribution of textual and temporal tokens of LLM-based MTSF methods, and we find that the temporal tokens in existing LLM-based methods cannot align well with the original textual tokens from LLMs [13, 16, 12, 14]. These observations inspire us to develop a Cross-modal LLM Fine-Tuning framework to consider the distribution discrepancy between textual and temporal input tokens.

Inspired by the above observations, we propose a **Cross-ModAl LLM Fine-Tuning (CALF)** framework, which employs cross-modal fine-tuning to allow more comprehensive alignment between temporal target modalities and textual source modalities. Specifically, CALF consists of two branches: the temporal target branch and the textual source branch. The temporal target branch processes time series information, while the textual source branch extracts and adapts information from pre-trained LLMs using aligned textual modal tokens. To bridge the modality gap between these branches, we introduce three meticulously designed cross-modal fine-tuning techniques (see Fig. 1b): (1) **Cross-Modal Match Module** integrates time series and textual inputs through principal word embedding extraction and a cross-attention mechanism, ensuring efficient alignment of the marginal input distribution between time series and text; (2) **Feature Regularization Loss** aligns the outputs of each intermediate layer, ensuring that gradients at every layer are more effectively guided for better weight updates; (3) **Output Consistency Loss** ensures that the output representations of textual and temporal series modalities correspond effectively, resolving discrepancies in the representation space and maintaining consistent semantic context for time series data. Through a more comprehensive alignment, our CALF consistently achieves state-of-the-art performance in both long-term and short-term forecasting across multiple datasets, demonstrating excellent few/zero-shot generalization capabilities, while maintaining significantly low complexity.

The contributions of this paper are threefold: (i) We identify the significant distribution discrepancies between textual and temporal modalities in existing LLM-based forecasting models and highlight the importance of addressing this misalignment for improved performance. (ii) We propose CALF, a novel framework that employs cross-modal fine-tuning techniques to comprehensively align temporal and textual data. The framework includes three specific methods: the Cross-Modal Match Module for aligning input distributions, Feature Regularization Loss for better gradient guidance and weight updates, and Output Consistency Loss for resolving output representation space discrepancies and maintaining consistent semantic context. (iii) Extensive experiments on eight real-world datasets

demonstrate that CALF achieves state-of-the-art performance on both long-term and short-term time series forecasting tasks, with favorable generalization ability and low computational complexity.

2 Related Work

2.1 Time Series Forecasting

In recent years, deep learning has significantly revolutionized the field of time series forecasting, with a plethora of methods emerging to enhance predictive accuracy [7, 8, 4, 17, 18, 19, 20]. Among these, Transformer-based models have emerged as the frontrunners, offering unparalleled performance due to their exceptional ability to model complex dependencies in data [6, 5, 21, 22, 9, 11, 20]. However, they often have limitations due to the scarcity of training data, overfitting in specific domains, and the necessity for intricate architectural designs.

In response to these challenges, the integration of LLMs into time series forecasting has emerged as a novel and promising direction. This approach leverages the extensive pre-training of LLMs to enhance the context-modeling capacity in time series analysis. A groundbreaking framework proposed by Zhou et al. [13] first demonstrated the potential of adapting LLMs for time series analysis. Following this paradigm, subsequent research has introduced further refinements and innovations. For example, Chang et al. [14] introduced a novel two-stage fine-tuning method and integrated time-series patching with additional temporal encoding into pre-trained LLMs. Cao et al. [15] incorporated decomposition of time series and selection-based prompts for adapting to non-stationary data. However, these works often directly input time series data into LLMs, overlooking the misalignment between time series and textual modalities. Some works have attempted to address this issue. Sun et al. [16] aligned time series data with LLM embeddings using contrastive learning and employed soft prompts for effective time series task handling. Jin et al. [12] reprogrammed time series input with text prototypes and enriches it using context as a prefix for LLM alignment. Despite these efforts, the alignment strategies have not been sufficiently effective.

2.2 Cross-Modal Fine-tuning

The objective of cross-modal fine-tuning is to apply models pre-trained on data-rich modalities to data-scarce modalities, addressing issues of data insufficiency and poor generalization [23]. Many existing works focus on transferring LLMs to other modalities, such as vision [24, 25], audio [26, 27], and biology [28, 29]. These efforts provide initial evidence of the cross-modal transfer capacity of pre-trained models. In the domain of time series, current research primarily leverages the powerful contextual modeling capabilities of LLMs to fine-tune them for improved forecasting performance [13, 12, 15, 14, 16], often neglecting the gap between the input and output distributions of language and time series modalities. In this work, we apply cross-modal fine-tuning techniques to address the challenge of transferring pre-trained language model knowledge to the time series modality.

3 Methodology

As shown in Fig. 2, our proposed CALF consists of two branches: the textual source branch and the temporal target branch. In concrete, the textual source branch takes the aligned text tokens X_{text} as input and employs L stacked pre-trained LLM layers to obtain the hidden text feature F_{text}^l , where $l = \{1, \dots, L\}$. A task-specific head is used to generate the output Y_{text} . Meanwhile, the temporal target branch works with the projected time series tokens X_{time} , and uses the same number of layers L with identical pre-trained weights as the textual source branch to obtain the hidden time feature F_{time}^l . The output of this branch is denoted as Y_{time} . To bridge the modality gap between these two branches, we utilize three cross-modal fine-tuning techniques to fine-tune the temporal target branch: the **Cross-Modal Match Module**, the **Feature Regularization Loss**, and the **Output Consistency Loss**. Detailed descriptions of these techniques will be provided in the following section.

3.1 Cross-Modal Match Module

As demonstrated in previous work [30], the matrices of word embedding layers in pre-trained LLMs constitute a well-structured context representation space, *e.g.*, semantic distances between different

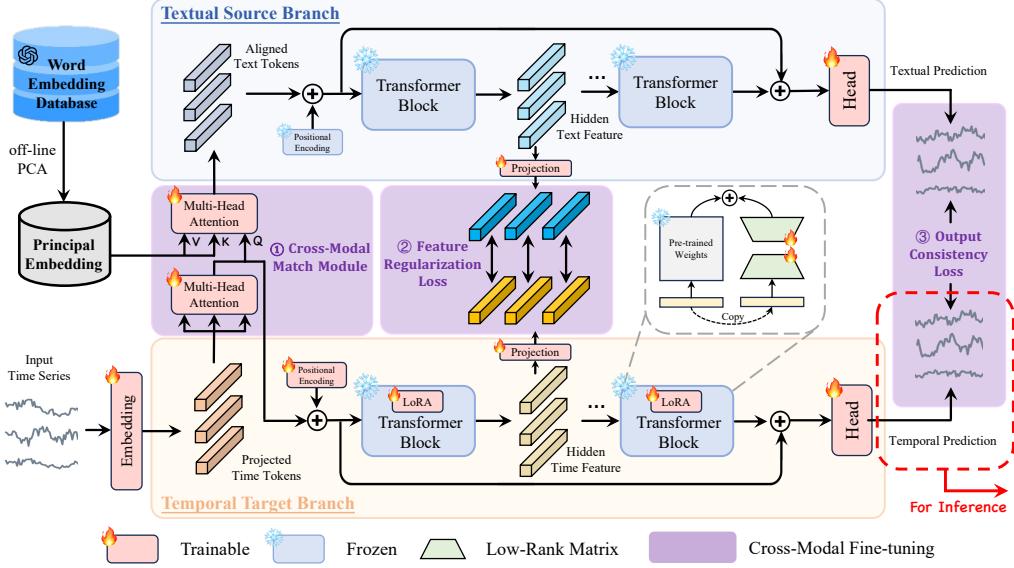


Figure 2: An overview of the proposed cross-modal fine-tuning framework. Above is the **Textual Source Branch**, and below is the **Temporal Target Branch**. To bridge the modality gap, the framework employs three **cross-modal fine-tuning** techniques: ①Cross-Modal Match Module, ②Feature Regularization Loss, and ③Output Consistency Loss.

words can be quantified through vector similarity. This word embedding layer represents the input distribution of the language modality in pre-trained LLMs. Despite this promising property, previous LLM-based time series methods often overlook this distribution, instead projecting the time series data to match the input dimensions of the language model [13, 15, 14].

In this work, we attempt to align the input distribution of time series with the word embedding of LLMs. Therefore, we propose a cross-modal match module to deal with this problem. Specifically, given a multivariate time series $I \in \mathbb{R}^{T \times C}$ as input, where T is the input sequence length and C is the number of variants, we first use the embedding layer similar to [31], followed by Multi-head Self Attention (MHSA) to get the projected time tokens X_{time} :

$$X_{time} = \text{MHSA}(\text{Embedding}(I)) \in \mathbb{R}^{C \times M}, \quad (1)$$

where M is the feature dimension of pre-trained LLMs. The embedding layer $\text{Embedding}(\cdot)$ performs a channel-wise dimensional mapping from T to M .

After that, we consider using cross-attention to align X_{time} from the temporal modality and the word embedding dictionaries $\mathcal{D} \in \mathbb{R}^{|\mathcal{A}| \times M}$, where $|\mathcal{A}|$ is the size of the alphabet, to the textual modality. However, due to $|\mathcal{A}|$ is usually huge, e.g., 50257 in GPT2 [32], directly using cross-attention incurs significant cost. Observing that semantic-similar words form ‘synonym clusters’, we propose a principal word embedding extraction strategy, which uses the cluster center to represent surrounding words, to reduce the number of word entries. Specifically, we use Principal Component Analysis (PCA) to perform dimension reduction on \mathcal{D} to obtain the principal word embeddings $\hat{\mathcal{D}} \in \mathbb{R}^{d \times M}$,

$$\hat{\mathcal{D}} = \text{PCA}(\mathcal{D}), \quad (2)$$

where d is a pre-defined low dimension and satisfies $d \ll |\mathcal{A}|$.

It is worth noting that this process needs to be done only once before model training and does not incur much training overhead. We then use Multi-head Cross-Attention with $\hat{\mathcal{D}}$ as key and value, and X_{time} as query to align the principal word embeddings and temporal tokens to obtain the aligned text tokens $X_{text} \in \mathbb{R}^{C \times M}$,

$$X_{text} = \text{Softmax}\left(\frac{QK^T}{\sqrt{C}}\right)V, \quad (3)$$

$$Q = X_{time}W_q, K = \hat{D}W_k, V = \hat{D}W_v,$$

where W_q , W_k and $W_v \in \mathbb{R}^{M \times M}$ are the projection matrices for the query (Q), key (K), and value (V), respectively.

3.2 Feature Regularization Loss

The pre-trained weights in LLMs are based on their original textual modality data. To more effectively adapt these pre-trained weights to time series data, we align the outputs of each intermediate layer in the temporal target branch with those of the textual source branch. This alignment process, facilitated by feature regularization loss, matches the intermediate features between two branches, allowing gradients at each intermediate layer to be more effectively guided for better weight updates. Formally, given F_{text}^l and F_{time}^l from the outputs of the l -th Transformer block in the textual source branch and temporal target branches, respectively, the feature regularization loss is defined as:

$$\mathcal{L}_{feature} = \sum_{i=1}^L \gamma^{(L-i)} \text{sim}(\phi_i^{text}(F_{text}^l), \phi_i^{time}(F_{time}^l)), \quad (4)$$

where γ is a hyper-parameter that controls the loss scale from different layers, and $\text{sim}(\cdot, \cdot)$ is a chosen similarity function, such as L_1 loss. Following [33], we introduce two trainable projection layers $\phi_l^{text}(\cdot)$ and $\phi_l^{time}(\cdot)$ to transform the features from textual and temporal modalities to the shared representation space.

3.3 Output Consistency Loss

Building on the feature regularization loss, we further ensure consistent semantic context between the textual and temporal modalities. Output consistency loss achieves this by ensuring that the output distributions correspond effectively, resolving discrepancies in the representation space. This alignment maintains a coherent and unified semantic representation for both the time series and textual data, facilitating more accurate and reliable model predictions. Specifically, given the outputs Y_{text} and Y_{time} from the textual source branch and temporal target branch respectively, the output consistency loss is defined as:

$$\mathcal{L}_{output} = \text{sim}(Y_{text}, Y_{time}). \quad (5)$$

3.4 Parameter Efficient Training

To avoid catastrophic forgetting and improve training efficiency, we employ the parameter-efficient training technique to fine-tune the pre-trained LLMs. Specifically, for the temporal target branch, we introduce Low-rank Adaptation (LoRA) [34] and fine-tune the positional encoding weights. The total loss during training is the weighted summation of the supervised loss \mathcal{L}_{sup} , the feature regularization loss $\mathcal{L}_{feature}$, and the output consistency loss \mathcal{L}_{output} :

$$\mathcal{L}_{total} = \mathcal{L}_{sup} + \lambda_1 \mathcal{L}_{feature} + \lambda_2 \mathcal{L}_{output}, \quad (6)$$

where λ_1 and λ_2 are hyper-parameters. In the inference stage, only the output of the temporal target branch will serve as the model output.

4 Experiments

To demonstrate the effectiveness of the proposed CALF, we conduct extensive experiments on various time series forecasting tasks, including long/short-term forecasting and few/zero-shot learning. Additionally, we validate the model with low complexity, highlighting its efficiency in practical applications.

Baselines. We carefully select representative baselines from the recent time series forecasting landscape, including the following categories: (1) LLMs-based models: TimeLLM [12] and GPT4TS [13]; (2) Transformer-based models: PatchTST [6], iTransformer [31], Crossformer [5], ETSformer [21], FEDformer [9] and Autoformer [22]; (3) CNN-based models: TCN [35], MICN [17] and TimesNet [4]; (4) MLP-based models: DLinear [7] and TiDE [8]. Besides, N-HiTS [36] and N-BEATS [37] are included for short-term forecasting.

Implementation Details. Following [13], we use pre-trained GPT2 based model [32] with the first 6 Transformer layers as our backbone. Optimization is conducted using the Adam optimizer [38], with a learning rate of 0.0005. For the total loss function, we set the hyper-parameters $\gamma = 0.8$, $\lambda_1 = 1$ and $\lambda_2 = 0.01$. In terms of loss functions for long-term forecasting, we apply L1 loss across all three loss types for ETT datasets, while for the other three datasets, smooth L1 loss is utilized. For short-term forecasting, we compute supervised loss with SMAPE, modal consistency loss with MASE, and feature regularization loss with smooth L1 loss, respectively. More details are provided in Appendix D.

4.1 Long-term Forecasting

Setups. We conduct experiments on seven widely-used real-world datasets, including the Electricity Transformer Temperature (ETT) dataset with its four subsets (ETTh1, ETTh2, ETTm1, ETTm2), Weather, Electricity, and Traffic [22]. Detailed descriptions of datasets are provided in Appendix C.1. The input time series length T is fixed as 96 for a fair comparison, and we adopt four distinct prediction horizons $H \in \{96, 192, 336, 720\}$. Consistent with prior works, the Mean Square Error (MSE) and Mean Absolute Error (MAE) are chosen as evaluation metrics.

Results. Comprehensive long-term forecasting results are presented in Tab. 1. Our method consistently delivers state-of-the-art performance, achieving the top results in 56 evaluations, in contrast to the nearest competing baseline which achieves top results only 7 times. Notably, our approach reduces MSE/MAE by 7.05%/6.53% compared to the state-of-the-art Transformer-based model PatchTST. In comparison with the LLM-powered method GPT4TS, we observe a reduction of 5.94%/5.14% in MSE/MAE. Moreover, our improvements are substantial against other baseline methods, exceeding 10% in most cases.

Models	CALF (Ours)	TimeLLM [†] [12]	GPT4TS [†] [13]	PatchTST [6]	iTransformer [31]	Crossformer [5]	FEDformer [9]	TimesNet [4]	MICN [17]	DLinear [7]	TiDE [8]
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
ETTm1	0.395 0.390	0.410 0.409	0.389 0.397	0.381 0.395	0.407 0.411	0.502 0.502	0.448 0.452	0.400 0.406	0.392 0.413	0.403 0.407	0.412 0.406
ETTm2	0.281 0.321	0.296 0.340	0.285 0.331	0.285 0.327	0.291 0.335	0.216 0.707	0.305 0.349	0.291 0.333	0.328 0.350	0.401 0.401	0.289 0.326
ETTh1	0.432 0.428	0.460 0.449	0.447 0.436	0.450 0.441	0.455 0.448	0.620 0.572	0.440 0.460	0.458 0.450	0.558 0.535	0.456 0.452	0.445 0.432
ETTh2	0.349 0.382	0.389 0.408	0.381 0.408	0.366 0.394	0.381 0.405	0.942 0.684	0.437 0.449	0.414 0.427	0.587 0.525	0.559 0.515	0.611 0.550
Weather	0.250 0.274	0.274 0.290	0.264 0.284	0.258 0.280	0.257 0.279	0.259 0.315	0.309 0.360	0.259 0.287	0.242 0.299	0.265 0.317	0.271 0.320
Electricity	0.175 0.265	0.223 0.309	0.205 0.290	0.216 0.304	0.178 0.270	0.244 0.334	0.214 0.327	0.192 0.295	0.186 0.294	0.212 0.300	0.251 0.344
Traffic	0.439 0.281	0.541 0.358	0.488 0.317	0.555 0.361	0.428 0.282	0.550 0.304	0.610 0.376	0.620 0.336	0.541 0.315	0.625 0.473	0.760 0.473

[†]We utilize their official codebase with the same experimental setup as ours, including input length and a GPT2 model with 6 layers, to ensure the fairness of the results. Other results are obtained from [31].

Table 1: Multivariate long-term forecasting results. The input sequence length T is set to 96 for all baselines. All the results are averaged from 4 different prediction lengths $H \in \{96, 192, 336, 720\}$. The best and second best results are in **bold** and underlined. Appendix F.1 shows the full results.

4.2 Short-term Forecasting

Setups. We adopt the M4 datasets [39], which comprise univariate marketing data collected yearly, quarterly, and monthly. Comprehensive details are available in Appendix C.2. In this case, the prediction horizons are comparatively short, ranging in [6, 48]. Correspondingly, the input lengths are set to be twice the size of the prediction horizons. The evaluation metrics are symmetric mean absolute percentage error (SMAPE), mean absolute scaled error (MSAE), and overall weighted average (OWA).

Results. As shown in Tab. 2, our method demonstrates superior performance in short-term forecasting across various evaluation metrics. Notably, it achieves the best results in 14 out of 15 categories, markedly outperforming all baselines. In comparison with TimesNet, currently the leading method in short-term forecasting, our model achieves a 1% overall improvement in performance.

Models	CALF (Ours)	TimeLLM [12]	GPT4TS [13]	PatchTST [6]	ETSformer [21]	FEDformer [9]	Autoformer [22]	TimesNet [4]	TCN [35]	N-HiTS [36]	N-BEATS [37]	DLinear [7]	
Yearly	SMAPE	13.351	13.419	13.531	13.477	18.009	13.728	13.974	13.387	14.920	13.418	13.436	16.965
	MASE	3.003	3.005	3.015	3.019	4.487	3.048	3.134	2.996	3.364	3.045	3.043	4.283
	OWA	0.786	0.789	0.793	<u>0.792</u>	1.115	0.803	0.822	0.786	0.880	0.793	0.794	1.058
Quarterly	SMAPE	9.990	10.110	10.177	10.380	13.376	10.792	11.338	10.100	11.122	10.202	10.124	12.145
	MASE	1.164	1.178	1.194	1.233	1.906	1.283	1.365	1.182	1.360	1.194	1.169	1.520
	OWA	0.878	0.889	0.898	0.921	1.302	0.958	1.012	0.890	1.001	0.899	<u>0.886</u>	1.106
Monthly	SMAPE	12.643	12.980	12.894	12.959	14.588	14.260	13.958	12.679	15.626	12.791	12.677	13.514
	MASE	0.922	0.963	0.956	0.970	1.368	1.102	1.103	<u>0.933</u>	1.274	0.969	<u>0.937</u>	1.037
	OWA	0.872	0.903	0.897	0.905	1.149	1.012	1.002	<u>0.878</u>	1.141	0.899	0.880	0.956
Others	SMAPE	4.552	4.795	4.940	4.952	7.267	4.954	5.485	4.891	7.186	5.061	4.925	6.709
	MASE	3.092	3.178	3.228	3.347	5.240	3.264	3.865	3.302	4.677	3.216	3.391	4.953
	OWA	0.967	<u>1.006</u>	1.029	1.049	1.591	1.036	1.187	1.035	1.494	1.040	1.053	1.487
Average	SMAPE	11.765	11.983	11.991	12.059	14.718	12.840	12.909	<u>11.829</u>	13.961	11.927	11.851	13.639
	MASE	1.567	1.595	1.600	1.623	2.408	1.701	1.771	<u>1.585</u>	1.945	1.613	1.599	2.095
	OWA	0.844	0.859	0.861	0.869	1.172	0.918	0.939	<u>0.851</u>	1.023	0.861	0.855	1.051

Table 2: Short-term forecasting results on M4 dataset. The input length and prediction length are set to [12, 96] and [6, 48], respectively. Appendix F.2 shows the full results.

4.3 Few/zero-shot Learning

LLMs have demonstrated remarkable performance in both few-shot and zero-shot tasks. The capabilities of few-shot and zero-shot learning are critically important for general time series forecasting models [40, 41, 42, 43]. To thoroughly assess the generalized ability of our method in time series forecasting, we conduct experiments under few-shot and zero-shot learning settings. In few-shot learning, only a small ratio of the training data is utilized. For zero-shot learning, the model trained on one dataset is directly employed for testing on another dataset without any additional training.

Few-shot Learning. We conduct few-shot experiments on four ETT datasets. Specifically, for each dataset, we utilize only the first 10% of the training data. This constrained data scenario presents a considerable challenge, testing the ability of the model to learn effectively with limited information. Tab. 3 demonstrates that our method outperforms other baselines, highlighting its robustness in the few-shot setting. Compared with GPT4TS and PatchTST, our method achieves an average reduction of 8% and 9%, respectively.

Models	CALF (Ours)	TimeLLM [12]	GPT4TS [13]	PatchTST [6]	Crossformer [5]	FEDformer [9]	TimesNet [4]	MICN [17]	DLinear [7]	TiDE [8]										
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE								
ETTm1	0.504	0.462	0.636	0.512	0.608	0.500	0.557	0.483	1.340	0.848	0.696	0.572	0.673	0.534	0.970	0.674	0.567	0.499	0.515	0.469
ETTm2	0.302	0.330	0.348	0.343	0.303	0.336	0.295	0.334	1.985	1.048	0.356	0.392	0.321	0.354	1.073	0.716	0.329	0.382	0.303	0.337
ETTh1	0.644	0.541	0.765	0.584	0.689	0.555	0.683	0.546	1.744	0.914	0.750	0.607	0.865	0.625	1.405	0.814	0.647	0.552	0.779	0.604
ETTh2	0.419	0.427	0.589	0.498	0.579	0.497	0.550	0.487	3.139	1.378	0.553	0.525	0.476	0.463	2.533	1.158	0.441	0.458	0.421	0.428

Table 3: Few-shot learning results on 10% training data of ETT datasets. All the results are averaged from 4 different prediction lengths $H \in \{96, 192, 336, 720\}$. Appendix F.3 shows the full results.

Zero-shot Learning. Going beyond few-shot scenarios, we further delve into zero-shot learning, where LLMs demonstrate their prowess as adept and intuitive reasoners. In this setting, models trained on one dataset ♦ are evaluated on an entirely different dataset ★, without any further training. As shown in Tab. 4, our method stands out for its exceptional performance, surpassing GPT4TS and PatchTST by 4% and 9% respectively. This indicates that our approach significantly enhances the model’s capability for effective learning transfer across different domains.

4.4 Efficiency Analysis

We conduct experiments on five datasets: ETTm1, ETTh1, ECL, Traffic, and Weather. The input and prediction lengths are both set to 96. As shown in Tab. 5, our proposed CALF shows significant

Models	CALF (Ours)		TimeLLM [12]		GPT4TS [13]		PatchTST [6]		Crossformer [5]		FEDformer [9]		TimesNet [4]		MICN [17]		DLinear [7]		TiDE [8]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
h1 → m1	0.755	0.574	0.847	0.565	0.798	0.574	0.894	0.610	0.999	0.736	0.765	0.588	0.794	0.575	1.439	0.780	0.760	0.577	0.774	0.574
h1 → m2	0.316	0.355	0.315	0.357	0.317	0.359	0.318	0.362	1.120	0.789	0.357	0.403	0.339	0.370	2.428	1.236	0.399	0.439	0.314	0.355
h2 → m1	0.836	0.586	0.868	0.595	0.920	0.610	0.871	0.596	1.195	0.711	0.741	0.588	1.286	0.705	0.764	0.601	0.778	0.594	0.841	0.590
h2 → m2	0.319	0.360	0.322	0.363	0.331	0.371	0.420	0.433	2.043	1.124	0.365	0.405	0.361	0.390	0.527	0.519	0.496	0.496	0.321	0.364

Table 4: Zero-shot learning results on ETT datasets, where ‘h1’, ‘h2’, ‘m1’, and ‘m2’ denote ETTh1, ETTh2, ETTm1, and ETTm2 respectively. “ $\diamond \rightarrow \star$ ” indicates that models trained on the dataset \diamond are evaluated on a distinct dataset \star . All the results are averaged from 4 different prediction lengths $H \in \{96, 192, 336, 720\}$. Appendix F.3 shows the full results.

	ETTm1	ETTh1	Time (s)		ECL	Traffic	Weather	ETTm1	ETTh1	MSE / MAE		ECL	Traffic	Weather
GPT4TS [13]	626	81	8274		15067	596	0.329 / 0.364	0.376 / 0.397	0.185 / 0.272	0.468 / 0.307		0.182 / 0.223		
Time-LLM [12]	1476	314	33209		62412	1262	0.359 / 0.381	0.398 / 0.410	0.204 / 0.293	0.536 / 0.359		0.195 / 0.233		
CALF (Ours)	135	27	251		614	123	0.323 / 0.349	0.369 / 0.389	0.145 / 0.238	0.407 / 0.268		0.164 / 0.204		

Table 5: Comparison of different LLM-based time series forecasting methods in terms of computation time and performance (MSE/MAE) across various datasets. The input and predict length are both set to 96.

improvements in both efficiency and accuracy compared with other LLM-based methods. We also provide theoretical complexity analysis for various Transformer-based methods in Appendix E.

5 Abaltion Study

Ablation on Different Loss Functions. The feature regularization loss $\mathcal{L}_{feature}$ aligns the intermediate features between the textual source branch and the temporal target branch, while the output consistency loss \mathcal{L}_{output} ensures output coherence across modalities. The supervised loss \mathcal{L}_{sup} directly guides learning with ground truth data. We analyze the specific effects of each proposed loss function as detailed in Tab. 6. Employing only the supervised loss resulted in MSE/MAE of 0.446/0.438 on ETTh1 and 0.263/0.286 on Weather, respectively. The addition of feature regularization loss $\mathcal{L}_{feature}$ or output consistency loss \mathcal{L}_{output} led to incremental improvements, with the best performance observed when all three losses were combined, achieving the lowest MSE and MAE on both datasets.

Ablation on the Number of Principal Components. We employ PCA to conduct dimensional reduction on the original word embeddings for efficient training. Despite the reduced cost, however, PCA may inevitably lead to information loss. In this section, we ablate the number of principal components d to present the effects. The experimental results are given in Fig. 3. It can be seen that the performance is not that sensitive to different numbers of principal components. In addition, a smaller d causes performance degradation due to the missing key information, while a larger d causes information redundancy which causes learning difficulty. In practice, we chose $d = 500$, which can attain an explainable variance ratio of 88% while achieving satisfactory performance.

$\mathcal{L}_{feature}$	\mathcal{L}_{output}	\mathcal{L}_{sup}	ETTh1	Weather
MSE	MAE	MSE	MAE	MAE
—	—	✓	0.446	0.438
✓	—	✓	0.434	0.431
—	✓	✓	0.438	0.426
✓	✓	✓	0.432	0.428
			0.250	0.274

Table 6: Ablation on different loss functions on ETTh1 and Weather datasets.

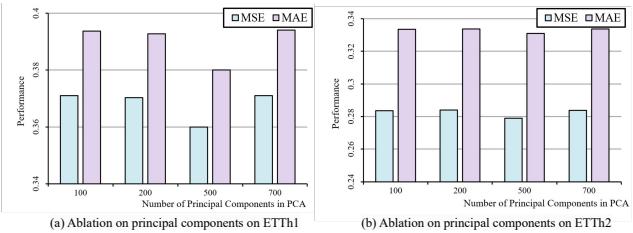


Figure 3: Ablation on different low dimension d of PCA on (a) ETTh1 and (b) ETTh2 datasets.

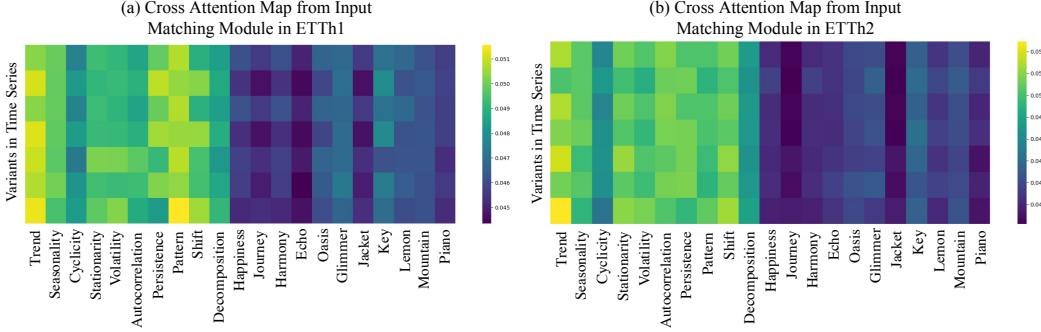


Figure 4: Cross-attention maps from the Cross-Modal Match Module for ETTh1 (left) and ETTh2 (right). Each row represents a time series instance, while columns correspond to selected words, including both time-related terms (*e.g.*, trend, seasonality) and general terms (*e.g.*, echo, key). Each cell indicates the relevance of the respective channel to the selected word.

6 Discussion

Difference form Other Work. One concurrent work [12] also considers cross attention to extracting knowledge from the word embedding layer, and we would like to clarify the difference to emphasize our contribution. First, the existing method uses cross-attention to generate embeddings and combines them with prompt prefixes as input to frozen LLMs, while our CALF aims to generate aligned textual tokens as the input of the textual modal branch for subsequent cross-modal distillation. Second, previous work introduces linear weight $W \in \mathbb{R}^{|\mathcal{A}| \times d}$ to learn text prototype during training. However, given the huge word space $|\mathcal{A}|$, this solution can lead to significant costs, while our approach uses an offline manner to generate synonym clusters, which guarantees efficiency.

Interpretability on Implicit Input Alignment. To narrow the temporal-textual modality gap, we perform cross-attention on word embedding weights to generate aligned text tokens instead of intuitive natural language. As shown in Fig. 4, we visualize the cross-attention maps from the Cross-Modal Match Module for the ETTh1 and ETTh2 datasets. Each row in the maps represents a time series instance, while columns correspond to selected words, including both time-related terms (*e.g.*, trend, seasonality) and general terms (*e.g.*, echo, key). Each cell indicates the relevance of the respective channel to the selected word. Our analysis reveals that the Cross-Modal Match Module effectively aligns time series tokens with word embeddings that describe temporal characteristics. The attention distributions show that time series data align well with relevant textual descriptions, indicating that our module successfully bridges the gap between temporal and textual modalities.

Limitations and Future Works. Our input alignment method relies on implicit alignment, which may not fully leverage the explicit textual reasoning capabilities inherent in LLMs [44]. Existing methods use explicit text merely as prior knowledge [12], missing opportunities for deeper integration. Future works should focus on seamlessly incorporating explicit textual information into time series analysis through improved pre-training techniques or advanced representation methods.

7 Conclusion

In this work, we propose CALF, a novel cross-modal fine-tuning framework that leverages the robust capabilities of Large Language Models (LLMs) for time series forecasting. CALF effectively bridges the distribution discrepancy between temporal data and the textual nature of LLMs through the Cross-Modal Match Module, Feature Regularization Loss, and Output Consistency Loss. Extensive experiments across several real-world datasets validate that CALF sets a new benchmark in both long- and short-term forecasting, demonstrating strong generalization and low computational complexity. To further understand the robustness of our framework, we provide a probabilistic analysis in Appendix B.

References

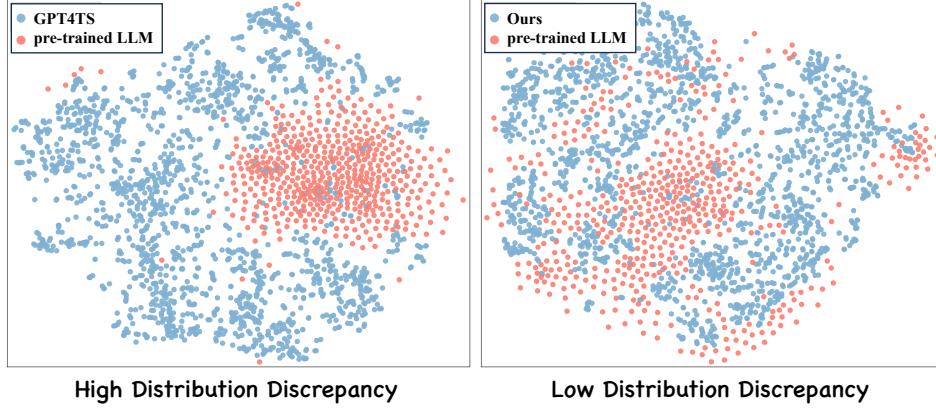
- [1] Rafal A Angryk, Petrus C Martens, Berkay Aydin, Dustin Kempton, Sushant S Mahajan, Sunitha Basodi, Azim Ahmadzadeh, Xumin Cai, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, et al. Multivariate time series dataset for space weather data analytics. *Scientific data*, 7(1):227, 2020. [1](#)
- [2] Ömer Fahrettin Demirel, Selim Zaim, Ahmet Çalışkan, and Pınar Özuyar. Forecasting natural gas consumption in istanbul using neural networks and multivariate time series methods. *Turkish Journal of Electrical Engineering and Computer Sciences*, 20(5):695–711, 2012. [1](#)
- [3] Andrew Patton. Copula methods for forecasting multivariate time series. *Handbook of economic forecasting*, 2:899–960, 2013. [1](#)
- [4] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023. [1, 3, 6, 7, 8, 17, 18, 19](#)
- [5] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 2023. [1, 3, 6, 7, 8, 16, 17, 18, 19](#)
- [6] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023. [1, 3, 6, 7, 8, 16, 17, 18, 19](#)
- [7] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023. [1, 3, 6, 7, 8, 17, 18, 19](#)
- [8] Abhimanyu Das, Weihao Kong, Andrew Leach, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023. [1, 3, 6, 7, 8, 17, 18, 19](#)
- [9] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pages 27268–27286. PMLR, 2022. [1, 3, 6, 7, 8, 16, 17, 18, 19](#)
- [10] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022. [1](#)
- [11] Tao Dai, Beiliang Wu, Peiyuan Liu, Naiqi Li, Jigang Bao, Yong Jiang, and Shu-Tao Xia. Periodicity decoupling framework for long-term series forecasting. In *International Conference on Learning Representations*, 2024. [1, 3](#)
- [12] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-LLM: Time series forecasting by reprogramming large language models. *International Conference on Learning Representations*, 2024. [1, 2, 3, 6, 7, 8, 9, 16, 17, 18, 19](#)
- [13] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One Fits All: Power general time series analysis by pretrained lm. *Advances in Neural Information Processing Systems*, 36, 2023. [1, 2, 3, 4, 6, 7, 8, 13, 16, 17, 18, 19](#)
- [14] Ching Chang, Wen-Chih Peng, and Tien-Fu Chen. LLM4TS: Two-stage fine-tuning for time-series forecasting with pre-trained llms. *arXiv preprint arXiv:2308.08469*, 2023. [2, 3, 4](#)
- [15] Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. TEMPO: Prompt-based generative pre-trained transformer for time series forecasting. *International Conference on Learning Representations*, 2024. [2, 3, 4](#)
- [16] Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. TEST: Text prototype aligned embedding to activate LLM’s ability for time series. In *The International Conference on Learning Representations*, 2024. [2, 3](#)

- [17] Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. MICN: Multi-scale local and global context modeling for long-term series forecasting. In *International Conference on Learning Representations*, 2022. 3, 6, 7, 8, 17, 18, 19
- [18] Peiyuan Liu, Beiliang Wu, Naiqi Li, Tao Dai, Fengmao Lei, Jigang Bao, Yong Jiang, and Shu-Tao Xia. WFTNet: Exploiting global and local periodicity in long-term time series forecasting. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023. 3
- [19] Tian Zhou, Ziqing Ma, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, Rong Jin, et al. FiLM: Frequency improved legendre memory model for long-term time series forecasting. *Advances in Neural Information Processing Systems*, 35:12677–12690, 2022. 3
- [20] Wang Xue, Tian Zhou, QingSong Wen, Jinyang Gao, Bolin Ding, and Rong Jin. CARD: Channel aligned robust blend transformer for time series forecasting. In *International Conference on Learning Representations*, 2024. 3
- [21] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven C. H. Hoi. ETS-former: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022. 3, 6, 7, 16, 17, 18
- [22] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, 2021. 3, 6, 7, 14, 16, 17, 18
- [23] Junhong Shen, Liam Li, Lucio M Dery, Corey Staten, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. Cross-modal fine-tuning: Align then refine. In *International Conference on Machine Learning*, pages 31030–31056. PMLR, 2023. 3
- [24] Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, Ethan Perez, and Davide Testuggine. Supervised multimodal bitransformers for classifying images and text. *arXiv preprint arXiv:1909.02950*, 2019. 3
- [25] Gaurav Verma, Minje Choi, Kartik Sharma, Jamelle Watson-Daniels, Sejoon Oh, and Srijan Kumar. Mysterious projections: Multimodal llms gain domain-specific visual capabilities without richer cross-modal projections. *arXiv preprint arXiv:2402.16832*, 2024. 3
- [26] Yufeng Jin, Guosheng Hu, Haonan Chen, Duoqian Miao, Liang Hu, and Cairong Zhao. Cross-modal distillation for speaker recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12977–12985, 2023. 3
- [27] Michael Hassid, Tal Remez, Tu Anh Nguyen, Itai Gat, Alexis Conneau, Felix Kreuk, Jade Copet, Alexandre Defossez, Gabriel Synnaeve, Emmanuel Dupoux, et al. Textually pretrained speech language models. *Advances in Neural Information Processing Systems*, 36, 2024. 3
- [28] Ria Vinod, Pin-Yu Chen, and Payel Das. Reprogramming pretrained language models for protein sequence representation learning. *arXiv preprint arXiv:2301.02120*, 2023. 3
- [29] Yijia Xiao, Jiezhong Qiu, Ziang Li, Chang-Yu Hsieh, and Jie Tang. Modeling protein using large-scale pretrain language model. *arXiv preprint arXiv:2108.07435*, 2021. 3
- [30] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 3
- [31] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. *International Conference on Learning Representations*, 2024. 4, 6, 16, 17
- [32] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. 2019. 4, 6, 16
- [33] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR, 2020. 5

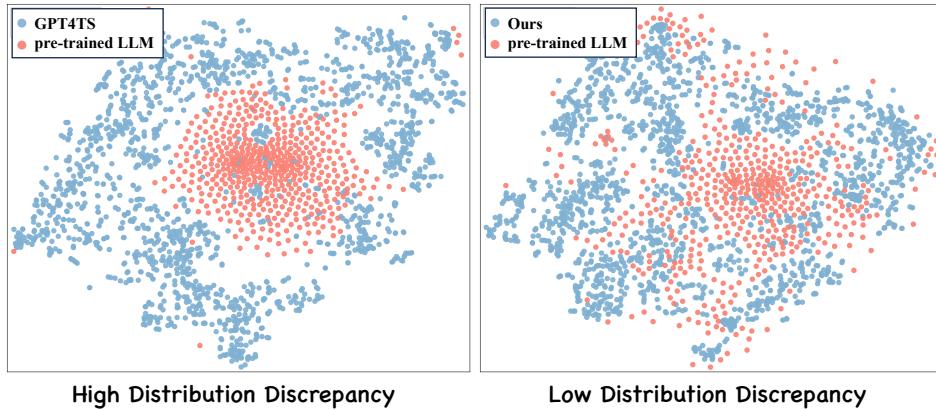
- [34] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 5, 16
- [35] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018. 6, 7, 17, 18
- [36] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. N-HiTs: Neural hierarchical interpolation for time series forecasting. *arXiv preprint arXiv:2201.12886*, 2022. 6, 7, 17, 18
- [37] Boris N Oreshkin, Dmitri Carpow, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *International Conference on Learning Representations*, 2019. 6, 7, 17, 18
- [38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6, 16
- [39] Spyros Makridakis. M4 dataset, 2018. 6
- [40] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020. 7
- [41] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 7
- [42] Xin Liu, Daniel McDuff, Geza Kovacs, Isaac Galatzer-Levy, Jacob Sunshine, Jiening Zhan, Ming-Zher Poh, Shun Liao, Paolo Di Achille, and Shwetak Patel. Large language models are few-shot health learners. *arXiv preprint arXiv:2305.15525*, 2023. 7
- [43] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, 2022. 7
- [44] Philipp Mondorf and Barbara Plank. Beyond accuracy: Evaluating the reasoning behavior of large language models—a survey. *arXiv preprint arXiv:2404.01869*, 2024. 9
- [45] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wanhai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021. 14, 16, 17
- [46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 16
- [47] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *ICLR*, 2022. 17, 18
- [48] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 1997. 17, 18

A Additional t-SNE Visualizations of Different Datasets

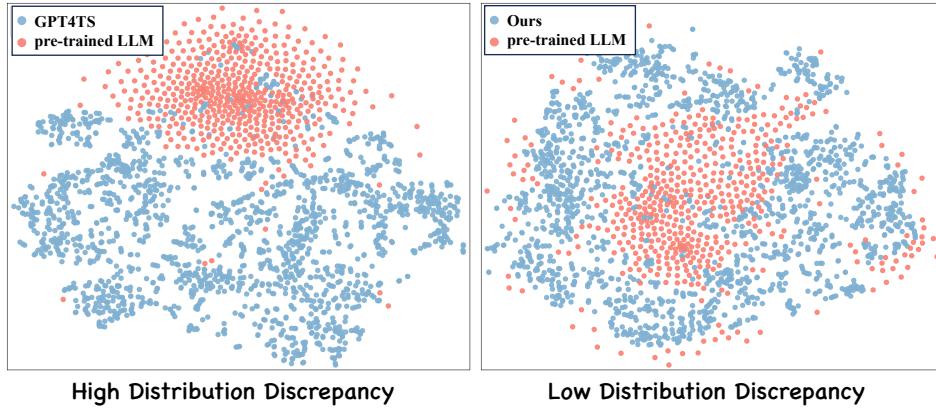
In addition to the ETTh2 dataset, we visualize three other datasets: Electricity, Weather, and Traffic, as shown in Fig. 5. The results for these datasets further demonstrate the effectiveness of our modality alignment approach. The t-SNE plots for these datasets exhibit similar cohesive integration, validating the robustness of our method across different data scenarios.



(a) The visualization of Weather dataset.



(b) The visualization of Electricity dataset.



(c) The visualization of Traffic dataset.

Figure 5: The t-SNE visualization of pre-trained word token embeddings of LLM with temporal tokens of (a) Weather, (b) Electricity, and (c) Traffic dataset from GPT4TS [13] (Left) and our method (Right).

B Probabilistic Analysis of Cross-modal Fine-tuning

To further explore the alignment between temporal and textual modalities in our proposed CALF framework, we adopt a probabilistic perspective rooted in transfer learning. This analysis provides a theoretical foundation for the cross-modal fine-tuning techniques employed in our model.

B.1 Probabilistic Framework

We define the temporal target domain and textual source domain as follows:

$$\mathcal{D}_T = \{p(X_T, y_T), P(y_T)\},$$

$$\mathcal{D}_S = \{p(X_S, y_S), P(y_S)\},$$

where X_T and X_S represent the input data, and y_T and y_S are the corresponding outputs for the temporal and textual domains, respectively. Using the Bayesian formula $p(X, y) = p(y | X)p(X)$, we can express the domains as:

$$\mathcal{D}_T = \{p(y_T | X_T)p(X_T), P(y_T)\},$$

$$\mathcal{D}_S = \{p(y_S | X_S)p(X_S), P(y_S)\}.$$

Here, $p(X)$ represents the input data distribution, $p(y | X)$ denotes the model, and $P(y)$ is the output distribution.

B.2 Cross-Modal Fine-Tuning Techniques

To address the alignment challenges between temporal and textual modalities, our CALF framework employs three cross-modal fine-tuning techniques, each corresponding to different components of the probabilistic framework: (1) **Cross-Modal Match Module** aligns the marginal input distributions $p(X_T)$ and $p(X_S)$, ensuring that the time series and text data have similar input distributions to facilitate better integration. (2) **Feature Regularization Loss** focuses on aligning the conditional probabilities $p(y_T | X_T)$ and $p(y_S | X_S)$, matching the intermediate features between the temporal and textual branches to improve model weight updates. (3) **Output Consistency Loss** addresses the alignment of the output distributions $P(y_T)$ and $P(y_S)$, ensuring that the final output representations from both modalities correspond effectively, maintaining a consistent semantic context for accurate predictions.

B.3 Theoretical Analysis

From a probabilistic perspective, our approach ensures comprehensive alignment across the entire data distribution, leading to better model generalization and performance. By addressing both the conditional and marginal distributions, our CALF framework effectively bridges the modality gap between temporal and textual data, thereby leveraging the full potential of pre-trained LLMs in time series forecasting. This analysis demonstrates the robustness and effectiveness of our framework in achieving state-of-the-art performance across various time series forecasting tasks.

C Dataset Details

C.1 Long-term Forecasting

We conduct extensive experiments on seven widely-utilized time series datasets for long-term forecasting. In line with the methodologies outlined in [45, 22], we chronologically partition each dataset into training, validation, and testing subsets. For the ETT dataset, we employ a 6:2:2 split ratio, whereas a 7:1:2 ratio is adopted for the remaining datasets. Detailed descriptions of these datasets are as follows:

- (1) **ETT**² (Electricity Transformer Temperature) dataset encompasses temperature and power load data from electricity transformers in two regions of China, spanning from 2016 to 2018. This dataset has two granularity levels: ETTh (hourly) and ETTm (15 minutes).

²<https://github.com/zhouhaoyi/ETDataset>

- (2) **Weather**³ dataset captures 21 distinct meteorological indicators in Germany, meticulously recorded at 10-minute intervals throughout 2020. Key indicators in this dataset include air temperature, visibility, among others, offering a comprehensive view of the weather dynamics.
- (3) **Electricity**⁴ dataset features hourly electricity consumption records in kilowatt-hours (kWh) for 321 clients. Sourced from the UCL Machine Learning Repository, this dataset covers the period from 2012 to 2014, providing valuable insights into consumer electricity usage patterns.
- (4) **Traffic**⁵ dataset includes data on hourly road occupancy rates, gathered by 862 detectors across the freeways of the San Francisco Bay area. This dataset, covering the years 2015 to 2016, offers a detailed snapshot of traffic flow and congestion.

We provide access to the ETT datasets through <https://github.com/zhouhaoyi/Informer2020>, while additional datasets are accessible at <https://github.com/thuml/Autoformer>. Detailed statistics for these datasets, including time steps, channels, and frequency, are presented in Tab. 7.

Datasets	Time steps	Channels	Frequency
Electricity	26304	321	1 hour
Weather	52696	21	10 min
Traffic	17544	862	1 hour
ETTm1	69680	7	15 min
ETTm2	69680	7	15 min
ETTh1	17420	7	1 hour
ETTh2	17420	7	1 hour

Table 7: The statistics of long-term forecasting datasets.

C.2 Short-term Forecasting

The M4 benchmark is an extensive assembly of 100,000 time series, sourced from a wide range of domains relevant to business, financial, and economic forecasting. These series are organized into six distinct datasets, with each dataset featuring sampling frequencies varying from yearly to hourly. We obtain the M4 dataset through <https://github.com/thuml/Time-Series-Library>. Detailed statistics for the M4 are presented in Tab. 8.

Datasets	Time steps	Frequency	Domains
M4-Yearly	23000	Yearly	Demographic
M4-Quarterly	24000	Quarterly	Finance
M4-Monthly	48000	Monthly	Industry
M4-Weekly	359	Weekly	Macro
M4-Daily	4227	Daily	Micro
M4-Hourly	414	Hourly	Other

Table 8: The statistics of short-term forecasting datasets.

³<https://www.bgc-jena.mpg.de/wetter>

⁴<https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014>

⁵<https://pems.dot.ca.gov>

C.3 Few/Zero-shot Learning

In our approach to few/zero-shot learning, we leverage the same four datasets from the ETT series as used in our long-term forecasting analysis, specifically ETTm1, ETTm2, ETTh1, and ETTh2.

D Implementation Details

Following [13], we utilize a pre-trained GPT2 based model [32], selecting the first 6 Transformer layers as our backbone. The model is fine-tuned using the LoRA method [34], with a rank setting of 8 and alpha set to 32. We also incorporate a dropout rate of 0.1 to enhance the model’s robustness. Optimization is achieved through the Adam optimizer [38], with a learning rate set at 0.0005. To tailor our model for specific forecasting tasks, we adjust the hyper-parameters of the total loss function to $\gamma = 0.8$, $\lambda_1 = 1$, and $\lambda_2 = 0.01$. For long-term forecasting loss functions, we apply L1 loss for all three types in the ETT datasets, while utilizing smooth L1 loss for the other datasets. For short-term forecasting, the model is refined using supervised loss with SMAPE, modal consistency loss with MASE, and feature regularization loss with smooth L1 loss. Additionally, we adopt a random seed of 2021 to ensure reproducibility. All our training processes are conducted on a single RTX 3090 GPU.

E Complexity Analysis

In Tab. 9, we present the theoretical computational complexity per layer for various Transformer-based models, including our proposed CALF. Unlike other Transformer-based approaches, whose computational complexities escalate with the increase in the input sequence length t , our CALF model, inspired by [31], primarily links its complexity to the number of channels C . This approach significantly reduces the overall complexity of our model compared to others.

Method	Encoder Complexity	Decoder Complexity
Transformer [46]	$O(T^2)$	$O(H(T + H))$
Informer [45]	$O(T \log T)$	$O(H(H + \log T))$
Autoformer [22]	$O(T \log T)$	$O((\frac{T}{2} + H) \log(\frac{T}{2} + H))$
FEDformer [9]	$O(T)$	$O(\frac{T}{2} + H)$
ETSformer [21]	$O(T \log T)$	$O(T \log H)$
Crossformer [5]	$O(\frac{C}{p^2} T^2)$	$O(\frac{C}{p^2} H(T + H))$
PatchTST [6]	$O((\frac{T}{p})^2)$	-
iTransformer [31]	$O(C^2)$	-
GPT4TS [13]	$O((\frac{T}{p})^2)$	-
Time-LLM [12]	$O((\frac{T}{p})^2)$	-
CALF (Ours)	$O(C^2)$	-

Table 9: Theoretical complexity per layer in Transformer-based models. T and H denote the length of the input and prediction sequence, respectively. C denotes the number of channels. p denotes the length of each patch in the patch-based methods.

F Full Results

F.1 Long-term Forecasting

Due to the limited space of the main text, we provide a more detailed comparison with additional baselines in Tab. 10, including LLM-based model (in yellow): TimeLLM [12] and GPT4TS [13]; Transformer-based models (in green): PatchTST [6], iTransformer [31], Crossformer [5],

FEDformer [9], Autoformer [22], and Informer [45]; CNN-based models (in purple): TimesNet [4] and MICN [17]; MLP-based models (in blue): DLinear [7] and TiDE [8].

Categories		LLM-based				Transformer-based						CNN-based			MLP-based	
Models	CALF (Ours)	TimeLLM [12]	GPT4TS [13]	PatchTST [6]	iTransformer [31]	Crossformer [5]	FEDformer [9]	Autoformer [22]	Informer [45]	TimesNet [4]	MICN [17]	DLinear [7]	TiDE [8]			
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.323 0.349	0.359 0.381	0.329 0.364	<u>0.321</u> <u>0.360</u>	0.341 0.376	0.360 0.401	0.379 0.419	0.505 0.475	0.672 0.571	0.338 0.375	0.316	0.362	0.345 0.372	0.352 0.373	
	192	0.374 0.375	0.383 0.393	0.368 <u>0.382</u>	0.362 0.384	0.382 0.395	0.402 0.440	0.426 0.441	0.553 0.496	0.795 0.669	0.374 0.387	<u>0.363</u>	0.390	0.380 0.389	0.389 0.391	
	336	0.409 0.399	0.416 0.414	<u>0.400</u> 0.403	0.392 <u>0.402</u>	0.418 0.418	0.543 0.528	0.445 0.459	0.621 0.537	1.212 0.871	0.410 0.411	0.408 0.426	0.413 0.413	0.423 0.413		
	720	0.477 <u>0.438</u>	0.483 0.449	<u>0.460</u> 0.439	0.450 0.435	0.487 0.456	0.704 0.642	0.543 0.490	0.671 0.561	1.166 0.823	0.478 0.450	0.481 0.476	0.474 0.453	0.485 0.448		
	Avg.	0.395 0.390	0.410 0.409	<u>0.389</u> 0.397	0.381 <u>0.395</u>	0.407 0.411	0.502 0.502	0.448 0.452	0.588 0.517	0.961 0.734	0.400 0.406	0.392 0.413	0.403 0.407	0.412 0.406		
ETTm2	96	0.178 0.256	0.193 0.280	0.178 0.263	0.178 <u>0.260</u>	0.185 0.272	0.273 0.356	0.203 0.287	0.255 0.339	0.365 0.453	0.187 0.267	<u>0.179</u>	0.275	0.193 0.292	0.181 0.264	
	192	0.242 0.297	0.257 0.318	<u>0.245</u> 0.306	0.249 0.307	0.253 0.313	0.426 0.487	0.269 0.328	0.249 0.309	0.281 0.340	0.533 0.563	0.307 0.376	0.284 0.362	0.246 <u>0.304</u>		
	336	0.307 0.339	0.317 0.353	<u>0.309</u> 0.347	0.313 0.346	0.315 0.350	1.013 0.714	0.325 0.366	0.339 0.372	1.363 0.887	0.321 0.351	0.325 0.388	0.369 0.427	0.307 <u>0.341</u>		
	720	0.397 0.393	0.419 0.411	0.409 0.408	<u>0.400</u> 0.398	0.413 0.406	3.154 1.274	0.421 0.415	0.433 0.432	3.379 1.338	0.408 0.403	0.502 0.490	0.554 0.522	0.407 <u>0.397</u>		
	Avg.	0.281 0.321	0.296 0.340	<u>0.285</u> 0.331	0.285 0.327	0.291 0.335	1.216 0.707	0.305 0.349	0.327 0.371	1.410 0.810	0.291 0.333	0.328 0.382	0.350 0.401	0.289 <u>0.326</u>		
ETTh1	96	0.369 0.389	0.398 0.410	<u>0.376</u> 0.397	0.393 0.408	0.386 0.404	0.420 0.439	<u>0.376</u> 0.419	0.449 0.459	0.865 0.713	0.384 0.402	0.421 0.431	0.386 0.400	0.384 <u>0.393</u>		
	192	<u>0.427</u> 0.423	0.451 0.440	0.438 0.426	0.445 0.434	0.441 0.436	0.540 0.519	0.420 0.448	0.436 0.429	0.500 0.482	1.008 0.792	0.474 0.487	0.437 0.432	0.436 0.422		
	336	0.456 0.436	0.508 0.471	0.479 0.446	0.484 0.451	0.489 0.461	0.722 0.648	<u>0.459</u> 0.465	0.521 0.496	1.107 0.809	0.491 0.469	0.569 0.551	0.481 0.459	0.480 <u>0.445</u>		
	720	0.479 0.467	0.483 0.478	0.495 0.476	<u>0.480</u> 0.471	0.508 0.493	0.799 0.685	0.506 0.507	0.514 0.512	1.181 0.865	0.521 0.500	0.770 0.672	0.519 0.516	0.481 <u>0.469</u>		
	Avg.	0.432 0.428	0.460 0.449	0.447 0.436	0.450 0.441	0.455 0.448	0.620 0.572	<u>0.440</u> 0.460	0.496 0.487	1.040 0.795	0.458 0.450	0.558 0.535	0.456 0.452	0.445 <u>0.432</u>		
ETTh2	96	0.279 0.331	0.295 0.346	0.295 0.348	<u>0.294</u> 0.343	0.300 0.349	0.745 0.584	0.358 0.397	0.346 0.388	3.755 1.525	0.340 0.374	0.299 0.364	0.333 0.387	0.400 0.440		
	192	0.353 0.380	0.386 0.399	0.386 0.404	<u>0.377</u> 0.393	0.379 0.398	0.877 0.656	0.429 0.439	0.456 0.452	5.602 1.931	0.402 0.414	0.441 0.454	0.477 0.476	0.528 0.509		
	336	0.362 0.394	0.447 0.443	0.421 0.435	<u>0.381</u> 0.409	0.418 0.429	1.043 0.731	0.496 0.487	0.482 0.486	4.721 1.835	0.452 0.452	0.654 0.567	0.594 0.541	0.643 0.571		
	720	0.404 0.426	0.428 0.444	0.422 0.445	<u>0.412</u> 0.433	0.428 0.445	1.104 0.763	0.463 0.474	0.515 0.511	3.647 1.625	0.462 0.468	0.956 0.716	0.831 0.657	0.874 0.679		
	Avg.	0.349 0.382	0.389 0.408	0.381 0.408	<u>0.366</u> 0.394	0.381 0.405	0.942 0.684	0.437 0.449	0.450 0.459	4.431 1.729	0.414 0.427	0.587 0.525	0.559 0.515	0.611 0.550		
Weather	96	0.164 0.204	0.195 0.233	0.182 0.223	0.177 0.218	0.174 <u>0.214</u>	0.158 0.230	0.217 0.296	0.266 0.336	0.300 0.384	0.172 0.220	<u>0.161</u>	0.229	0.196 0.255	0.202 0.261	
	192	<u>0.214</u> 0.250	0.240 0.269	0.231 0.263	0.225 0.259	0.221 <u>0.254</u>	0.206 0.277	0.276 0.336	0.307 0.367	0.598 0.544	0.219 0.261	0.220 0.281	0.237 0.296	0.242 0.298		
	336	0.269 0.291	0.293 0.306	0.283 0.300	0.278 0.297	0.278 <u>0.296</u>	<u>0.272</u> 0.335	0.339 0.380	0.359 0.395	0.578 0.523	0.280 0.306	0.278 0.331	0.283 0.335	0.287 0.335		
	720	0.355 0.352	0.368 0.354	0.360 0.350	0.354 <u>0.348</u>	0.358 <u>0.349</u>	0.398 0.418	0.403 0.428	0.419 0.428	1.059 0.741	0.365 0.359	0.311 0.356	<u>0.345</u> 0.381	0.351 0.386		
	Avg.	0.250 0.274	0.274 0.290	0.264 0.284	0.258 0.280	0.257 <u>0.279</u>	0.259 0.315	0.309 0.360	0.338 0.382	0.634 0.548	0.259 0.287	0.242 0.299	0.265 0.317	0.271 0.320		
Electricity	96	0.145 0.238	0.204 0.293	0.185 0.272	0.195 0.285	<u>0.148</u> 0.240	0.219 0.314	0.193 0.308	0.201 0.317	0.274 0.368	0.168 0.272	0.164 0.269	0.197 0.282	0.237 0.329		
	192	0.161 0.252	0.207 0.295	0.189 0.276	0.199 0.289	<u>0.162</u> 0.253	0.231 0.322	0.201 0.315	0.222 0.334	0.296 0.386	0.184 0.289	0.177 0.285	0.196 0.285	0.236 0.330		
	336	0.175 0.267	0.219 0.308	0.204 0.291	0.215 0.305	<u>0.178</u> 0.269	0.246 0.337	0.214 0.329	0.231 0.338	0.300 0.394	0.198 0.300	0.193 0.304	0.209 0.301	0.249 0.344		
	720	0.222 0.303	0.263 0.341	0.245 0.324	0.256 0.337	0.225 <u>0.317</u>	0.280 0.363	0.246 0.355	0.254 0.361	0.373 0.439	<u>0.220</u> 0.320	0.212 0.321	0.245 0.333	0.284 0.373		
	Avg.	0.175 0.265	0.223 0.309	0.205 0.290	0.216 0.304	<u>0.178</u> 0.270	0.244 0.334	0.214 0.327	0.227 0.338	0.311 0.397	0.192 0.295	0.186 0.294	0.212 0.300	0.251 0.344		
Traffic	96	0.407 0.268	0.536 0.359	0.468 0.307	0.544 0.359	0.395 0.268	0.522 <u>0.290</u>	0.587 0.366	0.613 0.388	0.719 0.391	0.593 0.321	0.519 0.309	0.650 0.396	0.805 0.493		
	192	0.430 0.278	0.530 0.354	0.476 0.311	0.540 0.354	0.417 0.276	0.530 0.293	0.604 0.373	0.616 0.382	0.696 0.379	0.617 0.336	0.537 0.315	0.598 0.370	0.756 0.474		
	336	0.444 0.281	0.530 0.349	<u>0.488</u> 0.317	0.551 0.358	0.433 0.283	0.558 0.305	0.621 0.383	0.622 0.337	0.777 0.420	0.629 0.336	0.534 0.313	0.605 0.373	0.762 0.477		
	720	0.477 0.300	0.569 0.371	<u>0.521</u> 0.333	0.586 0.375	0.467 0.302	0.589 0.328	0.626 0.382	0.660 0.408	0.864 0.472	0.640 0.350	0.577 0.325	0.645 0.394	0.719 0.449		
	Avg.	0.439 0.281	0.541 0.358	0.488 0.317	0.555 0.361	0.428 0.282	0.550 0.304	0.610 0.376	0.628 0.379	0.764 0.416	0.620 0.336	0.541 0.315	0.625 0.383	0.760 0.473		
1 st Count		50	0	1	<u>7</u>	<u>7</u>	2	1	0	0	0	4	0	0	2	

Table 10: Full results for long-term forecasting with different prediction lengths $H \in \{96, 192, 336, 720\}$. The input sequence length is set to 96 for all baselines. Avg. is averaged from all four prediction lengths. The best and the second best results are in **bold** and underlined. 1st Count indicates the number of times each method achieves the best results.

F.2 Short-term Forecasting

For short-term forecasting, a comparative analysis of our CALF model is presented against a range of baselines in Tab. 11. These include: GPT4TS [13], TimeLLM [12], PatchTST [6], ETSformer [21], FEDformer [9], Autoformer [22], TimesNet [4], TCN [35], N-HiTS [36], N-BEATS [37], DLinear [37], LSSL [47], and LSTM [48].

Models	CALF (Ours)	TimeLLM [12]	GPT4TS [13]	PatchTST [6]	ETSformer [21]	FEDformer [9]	Autoformer [22]	TimesNet [4]	TCN [35]	N-HiTS [36]	N-BEATS [37]	DLinear [7]	LSSL [47]	LSTM [48]	
Yearly	SMAPE	13.351	13.419	13.531	13.477	18.009	13.728	13.974	<u>13.387</u>	14.920	13.418	13.436	16.965	61.675	176.040
	MASE	<u>3.003</u>	3.005	3.015	3.019	4.487	3.048	3.134	2.996	3.364	3.045	3.043	4.283	19.953	31.033
	OWA	0.786	<u>0.789</u>	0.793	0.792	1.115	0.803	0.822	0.786	0.880	0.793	0.794	1.058	4.397	9.290
Quarterly	SMAPE	9.990	10.110	10.177	10.380	13.376	10.792	11.338	<u>10.100</u>	11.122	10.202	10.124	12.145	65.999	172.808
	MASE	1.164	1.178	1.194	1.233	1.906	1.283	1.365	1.182	1.360	1.194	<u>1.169</u>	1.520	17.662	19.753
	OWA	0.878	0.889	0.898	0.921	1.302	0.958	1.012	0.890	1.001	0.899	<u>0.886</u>	1.106	9.436	15.049
Monthly	SMAPE	12.643	12.980	12.894	12.959	14.588	14.260	13.958	12.679	15.626	12.791	<u>12.677</u>	13.514	64.664	143.237
	MASE	0.922	0.963	0.956	0.970	1.368	1.102	1.103	<u>0.933</u>	1.274	0.969	<u>0.937</u>	1.037	16.245	16.551
	OWA	0.872	0.903	0.897	0.905	1.149	1.012	1.002	<u>0.878</u>	1.141	0.899	0.880	0.956	9.879	12.747
Others	SMAPE	4.552	4.795	4.940	4.952	7.267	4.954	5.485	<u>4.891</u>	7.186	5.061	4.925	6.709	121.844	186.282
	MASE	3.092	<u>3.178</u>	3.228	3.347	5.240	3.264	3.865	3.302	4.677	3.216	3.391	4.953	91.650	119.294
	OWA	0.967	<u>1.006</u>	1.029	1.049	1.591	1.036	1.187	1.035	1.494	1.040	1.053	1.487	27.273	38.411
Average	SMAPE	11.765	11.983	11.991	12.059	14.718	12.840	12.909	<u>11.829</u>	13.961	11.927	11.851	13.639	67.156	160.031
	MASE	1.567	1.595	1.600	1.623	2.408	1.701	1.771	<u>1.585</u>	1.945	1.613	1.599	2.095	21.208	25.788
	OWA	0.844	0.859	0.861	0.869	1.172	0.918	0.939	<u>0.851</u>	1.023	0.861	0.855	1.051	8.021	12.642
$1^{\text{st}} \text{ Count}$		14	0	0	0	0	0	0	<u>2</u>	0	0	0	0	0	0

Table 11: Full results for short-term forecasting on M4 dataset. The input length and prediction length are set to [12, 96] and [6, 48], respectively. Average is the weighted average results from several datasets under different sample intervals. The best and the second best results are in **bold** and underlined. $1^{\text{st}} \text{ Count}$ indicates the number of times each method achieves the best results.

E.3 Few/Zero-shot Learning

We present the complete results of all prediction lengths $H \in \{96, 192, 336, 720\}$ for few-shot and zero-shot learning in Tab. 12 and Tab. 13, respectively.

Models	CALF (Ours)	TimeLLM [12]	GPT4TS [13]	PatchTST [6]	Crossformer [5]	FEDformer [9]	TimesNet [4]	MICN [17]	DLinear [7]	TiDE [8]					
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	
ETTm1	96	0.468	<u>0.445</u>	0.587	0.491	0.615	0.497	0.558	0.478	1.037	0.705	0.604	0.530	0.583	0.503
	192	0.479	<u>0.446</u>	0.606	0.490	0.597	0.492	0.539	0.471	1.170	0.778	0.641	0.546	0.608	0.515
	336	0.499	<u>0.463</u>	0.719	0.555	0.597	0.501	0.558	0.488	1.463	0.913	0.768	0.606	0.733	0.572
	720	<u>0.572</u>	<u>0.496</u>	0.632	0.514	0.623	0.513	0.574	0.498	1.693	0.997	0.771	0.606	0.768	0.548
Avg.		0.504	<u>0.462</u>	0.636	0.512	0.608	0.500	0.557	0.483	1.340	0.848	0.696	0.572	0.673	0.534
ETTm2	96	0.190	<u>0.268</u>	<u>0.189</u>	0.270	0.187	<u>0.266</u>	<u>0.189</u>	<u>0.268</u>	1.397	0.866	0.222	0.314	0.214	0.288
	192	0.257	0.311	0.264	0.319	<u>0.253</u>	<u>0.308</u>	0.248	<u>0.307</u>	1.757	0.987	0.284	0.351	0.271	0.325
	336	0.323	<u>0.334</u>	0.327	0.358	0.332	0.353	0.311	<u>0.346</u>	2.075	1.086	0.392	0.419	0.329	0.356
	720	0.441	0.410	0.454	0.428	<u>0.438</u>	<u>0.417</u>	0.435	0.418	2.712	1.253	0.527	0.485	0.473	0.448
Avg.		<u>0.302</u>	<u>0.330</u>	0.308	0.343	0.303	0.336	0.295	<u>0.334</u>	1.985	1.048	0.356	0.392	0.321	0.354
ETTh1	96	0.468	0.457	0.500	0.464	<u>0.462</u>	<u>0.449</u>	0.433	<u>0.428</u>	1.129	0.775	0.651	0.563	0.855	0.625
	192	<u>0.550</u>	0.501	0.590	0.516	<u>0.551</u>	<u>0.495</u>	0.509	<u>0.474</u>	1.832	0.922	0.666	0.562	0.791	0.589
	336	<u>0.581</u>	<u>0.521</u>	0.638	0.542	0.630	0.539	0.572	<u>0.509</u>	2.022	0.973	0.767	0.602	0.939	0.648
	720	0.978	0.685	1.334	0.816	1.113	0.738	1.221	0.773	1.903	0.986	0.918	0.703	<u>0.876</u>	<u>0.641</u>
Avg.		0.644	<u>0.541</u>	0.765	0.584	0.689	0.555	0.683	0.645	1.744	0.914	0.750	0.607	0.865	0.625
ETTh2	96	0.314	0.360	0.329	0.365	<u>0.327</u>	<u>0.359</u>	0.314	<u>0.354</u>	2.482	1.206	0.359	0.404	0.372	0.405
	192	<u>0.404</u>	<u>0.411</u>	0.414	0.413	0.403	<u>0.405</u>	0.420	0.415	3.136	1.372	0.460	0.461	0.483	0.463
	336	<u>0.458</u>	<u>0.452</u>	0.579	0.506	0.568	0.499	0.543	0.489	2.925	1.331	0.569	0.530	0.541	0.496
	720	0.502	0.487	1.034	0.711	1.020	0.725	0.926	0.691	4.014	1.603	0.827	0.707	0.510	0.491
Avg.		0.419	<u>0.427</u>	0.589	0.498	0.579	0.497	0.550	0.487	3.139	1.378	0.553	0.525	0.476	0.463
$1^{\text{st}} \text{ Count}$		16	0	4	<u>13</u>	0	0	0	0	0	0	0	4	4	

Table 12: Full results for few-shot learning on 10% training data of ETT datasets with different prediction lengths $H \in \{96, 192, 336, 720\}$. The input sequence length is set to 96 for all baselines. Avg. is averaged from all four prediction lengths. The best and the second best results are in **bold** and underlined. $1^{\text{st}} \text{ Count}$ indicates the number of times each method achieves the best results.

Models	CALF (Ours)	TimeLLM [12]		GPT4TS [13]		PatchTST [6]		Crossformer [5]		FEDformer [9]		TimesNet [4]		MICN [17]		DLinear [7]		TiDE [8]	
	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
h1 m1	96	0.767	0.564	0.804	0.565	0.809	0.563	0.908	0.596	0.856	0.649	0.731	0.561	0.764	0.563	0.832	0.621	<u>0.735</u>	<u>0.554</u>
	192	0.753	0.570	0.827	0.593	0.799	<u>0.567</u>	0.927	0.616	0.906	0.684	0.746	0.573	0.798	0.562	1.288	0.854	<u>0.752</u>	0.570
	336	0.745	0.575	0.835	0.600	0.803	<u>0.577</u>	0.920	0.621	1.104	0.796	0.775	0.596	0.790	0.584	1.721	0.972	<u>0.749</u>	0.579
	720	0.758	<u>0.590</u>	0.922	0.644	<u>0.783</u>	0.589	0.822	0.608	1.131	0.816	0.808	0.625	0.827	0.594	1.915	1.036	0.805	0.606
Avg.		0.755	0.574	0.847	0.600	0.798	0.574	0.894	0.610	0.999	0.736	0.765	0.588	0.794	<u>0.575</u>	1.439	0.870	<u>0.760</u>	0.577
h1 m2	96	0.218	0.301	0.212	0.298	<u>0.218</u>	0.304	0.219	0.305	0.611	0.588	0.257	0.345	0.245	0.322	0.496	0.556	0.239	0.343
	192	<u>0.278</u>	0.334	0.277	0.338	0.279	0.338	0.280	0.341	0.789	0.685	0.318	0.380	0.293	0.346	1.798	1.137	0.320	0.397
	336	0.338	0.369	0.336	0.371	0.342	0.376	0.341	0.376	1.469	0.927	0.375	0.417	0.361	0.382	2.929	1.472	0.409	0.453
	720	<u>0.431</u>	0.418	0.435	0.424	<u>0.431</u>	0.419	0.432	0.426	1.612	0.957	0.480	0.472	0.460	0.432	4.489	1.782	0.629	0.565
Avg.		0.316	0.355	<u>0.315</u>	<u>0.357</u>	0.317	0.359	0.318	0.362	1.120	0.789	0.357	0.403	0.339	0.370	2.428	1.236	0.399	0.439
h2 m1	96	0.897	0.589	0.891	0.587	0.985	0.604	0.815	0.560	1.032	0.620	0.734	0.578	1.205	0.678	<u>0.743</u>	0.577	0.762	<u>0.567</u>
	192	0.864	<u>0.584</u>	0.850	0.583	0.872	0.600	0.900	0.606	1.176	0.676	0.723	0.594	1.159	0.670	<u>0.750</u>	0.588	0.785	0.588
	336	0.816	0.585	0.853	0.594	0.926	0.614	0.906	0.602	1.199	0.718	0.750	0.590	1.197	0.689	<u>0.764</u>	0.606	0.767	0.594
	720	<u>0.768</u>	0.589	0.879	0.616	0.899	0.624	0.866	0.619	1.373	0.832	0.760	0.592	1.583	0.784	0.801	0.634	0.800	0.627
Avg.		0.836	0.586	0.868	0.595	0.920	0.610	0.871	0.596	1.195	0.711	0.741	<u>0.588</u>	1.286	0.705	<u>0.764</u>	0.601	0.778	0.594
h2 m2	96	0.225	0.310	0.228	<u>0.311</u>	0.235	0.316	0.288	0.345	0.821	0.634	0.261	0.347	0.244	0.324	0.327	0.414	0.264	0.366
	192	0.283	<u>0.342</u>	0.283	0.341	<u>0.287</u>	0.346	0.344	0.375	1.732	1.018	0.313	0.370	0.331	0.374	0.450	0.485	0.394	0.452
	336	<u>0.340</u>	<u>0.373</u>	0.343	0.376	0.361	0.391	0.438	0.425	2.587	1.393	0.401	0.431	0.386	0.405	0.526	0.526	0.506	0.513
	720	0.429	0.418	0.437	0.424	0.444	0.433	0.611	0.588	3.034	1.452	0.487	0.472	0.485	0.458	0.806	0.652	0.822	0.655
Avg.		0.319	0.360	0.322	<u>0.363</u>	0.331	0.371	0.420	0.433	2.043	1.124	0.365	0.405	0.361	0.390	0.527	0.519	0.496	0.496
1 st Count		19	7	2	1	0	7	1	0	0	0	9							

Table 13: Full results for zero-shot learning on ETT datasets with different prediction lengths $H \in \{96, 192, 336, 720\}$, where ‘h1’, ‘h2’, ‘m1’, and ‘m2’ denote ETTh1, ETTh2, ETTm1, and ETTm2 respectively.. The input sequence length is set to 96 for all baselines. “ $\diamond \rightarrow \star$ ” indicates that models trained on the dataset \diamond are evaluated on a distinct dataset \star . Avg. is averaged from all four prediction lengths. The best and the second best results are in **bold** and underlined. 1st Count indicates the number of times each method achieves the best results.

G Broader Impacts

Our work on the CALF framework for time series forecasting primarily focuses on enhancing predictive accuracy and generalization. While the positive societal impacts include improved forecasting for critical applications such as weather prediction, energy management, and financial modeling, potential negative impacts should be considered. These may include privacy concerns related to the data used for training and potential biases in predictions that could affect specific groups unfairly. To mitigate these risks, we advocate for careful data handling practices, transparency in model training, and ongoing monitoring to ensure fairness and accuracy in real-world applications.