AutoTimes: Autoregressive Time Series Forecasters via Large Language Models

Yong Liu[*], **Guo Qin**[*], **Xiangdong Huang, Jianmin Wang, Mingsheng Long**[✉]
School of Software, BNRist, Tsinghua University, China
{liuyong21,qinguo24}@mails.tsinghua.edu.cn,
{huangxdong,jimwang,mingsheng}@tsinghua.edu.cn

Abstract

Foundation models of time series have not been fully developed due to the limited availability of time series corpora and the underexploration of scalable pre-training. Based on the similar sequential formulation of time series and natural language, increasing research demonstrates the feasibility of leveraging large language models (LLM) for time series. Nevertheless, the inherent autoregressive property and decoder-only architecture of LLMs have not been fully considered, resulting in insufficient utilization of LLM abilities. To fully revitalize the general-purpose token transition and multi-step generation capability of large language models, we propose **AutoTimes** to repurpose LLMs as **Auto**regressive **Time s**eries forecasters, which projects time series into the embedding space of language tokens and autoregressively generates future predictions with arbitrary lengths. Compatible with any decoder-only LLMs, the consequent forecaster exhibits the flexibility of the lookback length and scalability with larger LLMs. Further, we formulate time series as prompts, extending the context for prediction beyond the lookback window, termed **in-context forecasting**. By introducing LLM-embedded textual timestamps, AutoTimes can utilize chronological information to align multivariate time series. Empirically, AutoTimes achieves state-of-the-art with 0.1% trainable parameters and over $5\times$ training/inference speedup compared to advanced LLM-based forecasters. Code is available at this repository: https://github.com/thuml/AutoTimes.

1 Introduction

Time series forecasting is of crucial demand in real-world applications, covering various domains including climate, economics, energy, operations, etc. [22, 43]. The growing challenges of general-purpose forecasting, where one model is versatile to handle variable-length scenarios [24, 41] and the prediction is necessarily instructed by auxiliary information in other modalities [40, 44], underscore the demand for foundation models [3] of time series, which are aimed to exhibit enhanced capabilities, including multi-step generation, zero-shot generalization [49, 13], in-context learning and multimodal utilization [15], thereby expanding the scope of time series forecasting to a wider range of situations.

Nevertheless, the development of time series foundation models has been hampered by the limited availability of large-scale pre-training datasets and the technical uncertainty of scalable backbones. In contrast, rapid progress is witnessed in large language models (LLM), facilitated by extensive text corpora [50], available pre-trained models [36], and well-established adaptation techniques [14]. Notably, language and time series share basic commonalities in sequence modeling and generation by learned token transitions, presenting opportunities to adopt off-the-shelf LLMs for time series.

[*]Equal Contribution

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

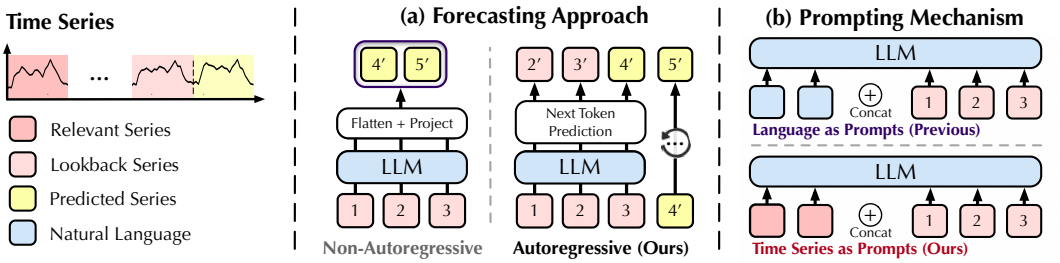

Figure 1: (a) Prevalent LLM4TS methods non-autoregressively generate predictions with the globally flattened representation of lookback series, while large language models inherently predict the next tokens by autoregression [47]. (b) Previous methods adopt language prompts that may lead to the modality disparity, while we find time series can be self-prompted, termed *in-context forecasting*.

Despite recent studies on large language models for time series (LLM4TS) achieving performance breakthroughs in current forecasting benchmarks [15], the mechanism by which LLMs are aligned to the time series modality still remains obscure. The pilot work, FPT [49] leverages LLMs as generic sequential representation extractors for time series, influencing subsequent LLM4TS methodologies. As depicted in Figure 1 (a), the non-autoregressive approach, where time series are segmented into tokens, flattens and projects all lookback tokens for the prediction in a single step. However, it causes inconsistencies in both model structure and generative approach of LLMs: *decoder-only models for autoregressive generation are converted to encoder-only and non-autoregressive forecasters*.

Given that prior studies [9, 38] reveal that generalization performance of LLMs is largely derived from the decoder-only structure trained autoregressively, talents of LLMs may not be fully exhibited. It is also supported by the recent rethinking of previous LLM4TS methods [35], which generally lack the maintenance of autoregression, the essential characteristic of both large language models and statistical forecasters [5, 39]. Therefore, autoregressive LLM4TS methods are underexplored, which can potentially unlock multi-step generation like LLMs, presenting one model for arbitrary lengths.

Motivated by the reflections, we propose **AutoTimes** to adapt LLMs as time series forecasters, which *retrieves the consistency of autoregression with revitalized LLM capabilities to produce foundation models for time series forecasting*. Technically, we independently embed time series segments into the latent space of language models by the consistent training objective: next token prediction [2]. To fully leverage the inherent token transitions of LLMs and reduce the training cost, we freeze the LLM and establish token embedding and projection for time series, which only account for up to 0.1% total parameters. The consequent forecaster adopts autoregressive inference like LLMs, which is no longer constrained to specific lookback/forecast lengths. Going beyond conventional time series forecasting, we propose **in-context forecasting** as shown in Figure 1, where time series can be self-prompted by relevant contexts. We further adopt LLM-embedded timestamps as the position embedding to utilize chronological information and align multiple variates. Our contributions are summarized as follows:

- By refining the inconsistency of non-autoregressive LLM4TS methods, we propose to inherit the autoregressive property of LLMs, which frees our method from training respectively on the lookback length and allows arbitrary-length predictions with chronological awareness.
- We present AutoTimes, a simple but effective approach to acquire LLM-based forecasters by lightweight adaptation, which utilizes the inherent token transition as the future extrapolation of time series. Further, we propose in-context forecasting, which renovates the conventional paradigm by introducing relevant time series prompts to enhance forecasting.
- Compared with state-of-the-art methods, our repurposed forecaster achieves superior performance while saving over 80% training and inference time, and further exhibits zero-shot generalizability, in-context forecasting, and scaling behavior empowered by LLMs.

2 Related Work

2.1 Autoregressive Models

Autoregression is an essential concept of both language modeling and time series forecasting. Despite prevalent deep forecasters [10, 26, 42, 48] adopt a non-autoregressive approach without the

requirement of iterative forecasting, autoregression, the absent exploration in deep forecasters, serves as the fundamental principle of statistical methods, which enables variable-length predictions. The most well-known model, ARIMA [4] is developed by incorporating differencing on AR and MA models, which are both autoregressive models with learned time-invariant transition from the past to the future. Incorporated with decomposition and pre-defined transitions, exponential smoothing [39] and state space models (SSM) [12, 23] also take the same autoregressive formulation.

Autoregressive language models [27, 31] are trained with fine-grained supervision, where the generated token of each position is independently supervised. Consequently, they are not constrained by specific input/output lengths and excel at multi-step generation. Furthermore, existing LLMs are inherently autoregressive models [47], which demonstrate advanced abilities that are not present in small models, such as the generalization [38], scalability [6], and task generality [31, 32]. Therefore, it is imperative to adapt off-the-shelf LLMs as autoregressive forecasters, which keeps the consistency to fully revitalize the model capacity and general-purpose token transitions.

2.2 Large Language Models for Time Series

With the immense advancement of large language model infrastructure, LLM4TS methods have been experiencing significant development in recent years. PromptCast [44] reformulates time series as text pairs and accomplishes forecasting as a sentence-to-sentence task. LLMTime [13] regards time series as numerical tokens, demonstrating the zero-shot generalizability in time series forecasting. FPT [49] fine-tunes parameters of the LLM to adapt it as a general representation extractor serving for multiple time series analysis tasks. UniTime [21] adapts a language model across diverse time series for a unified forecaster of multiple domains. Based on thriving prompting techniques, deft language prompts [15, 21] and soft prompting [7] for time series are further investigated.

LLM4TS methods have achieved performance breakthroughs in time series forecasting, but the cost of training and inference can sometimes be resource-consuming due to the immensity of LLMs. Recent revisiting of LLM4TS methods has revealed the inefficacy of LLMs adapted in the non-autoregressive approach [35]. By contrast, AutoTimes frozen LLMs, transfers the general-purpose token transition, and introduces minimal parameters to realize autoregressive next token prediction, thereby achieving better model efficiency and consistent utilization of large models. We further provide Table 1 that categorizes prevalent LLM4TS methods by several essential aspects.

2.3 Multimodal Language Models

Multimodal models have been well-developed upon LLMs, among which vision language models (VLM) have experienced rapid growth [1, 27]. The booming pre-trained vision backbones [11, 30], together with the instruction tuning paradigm, has revealed the potential of LLMs for vision tasks, where visual tokens and language tokens are concatenated as the input of the LLM [19, 20]. Inspired by this, previous LLM4TS methods utilize instructive language tokens as prefix-prompts for time series analysis [15, 34, 44]. Unlike previous works, our proposed method regards time series itself as the instructive prompt. It avoids the modality gap caused by concatenating time series and language tokens directly. We incorporate chronological information, the textual timestamp of time series, such that the language model can effectively perceive date and periodicity as the position embedding, and align simultaneous events from different time series [22] for multivariate forecasting.

Table 1: Comparison of LLM4TS methods: *Autoregressive* categories LLM-based forecasters by whether to conduct autoregression. *Freeze LLM* enables quick adaptation, which would otherwise require significant resources for fine-tuning. *Multimodal* refers to the utilization of information from other modalities. Prior to AutoTimes, none of the LLM4TS methods achieved all three.

Method	**AutoTimes**	TimeLLM [15]	UniTime [21]	FPT [49]	LLMTime [13]	TEST [34]	TEMPO [7]	PromptCast [44]
Autoregressive	✓	✗	✗	✗	✓	✗	✗	✗
Freeze LLM	✓	✓	✗	✗	✓	✓	✗	✓
Multimodal	✓	✓	✓	✗	✗	✓	✓	✓

3 Method

The proposed AutoTimes adapts large language models for multivariate time series forecasting. Given lookback observations $\mathbf{x}_{1:L} = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\} \in \mathbb{R}^{L \times C}$ with L time steps and C variates, the objective

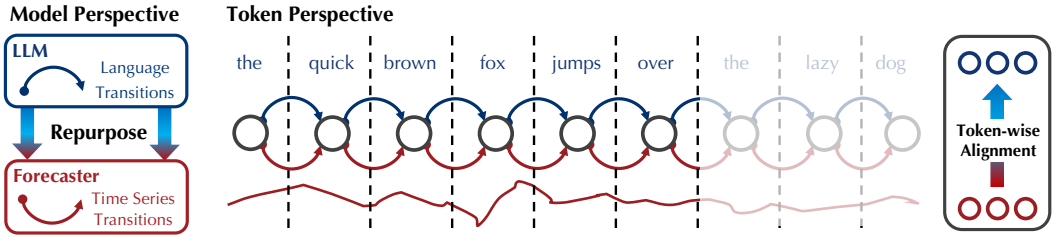

Figure 2: An example to illustrate how AutoTimes adapts language models for time series forecasting.

is to predict the future F time steps $\mathbf{x}_{L+1:L+F} = \{\mathbf{x}_{L+1}, \ldots, \mathbf{x}_{L+F}\} \in \mathbb{R}^{F \times C}$. Besides, the textual timestamp $\mathbf{a}_t$ (e.g. 2016/07/05 00:00:00), as the most common covariate, is adopted for prediction, which is aligned with time points $\mathbf{x}_t \in \mathbb{R}^C$ at time t. The task is to train an LLM-based forecaster f that is able to predict with the (varying) lookback length L for the (arbitrary) forecast length F as:

$$f : (\mathbf{x}_{1:L}, \mathbf{a}_{1:L+F}) \mapsto \hat{\mathbf{x}}_{L+1:L+F}. \tag{1}$$

3.1 Modality Alignment

Time series token To empower the forecaster with the capability to predict time series for arbitrary lengths, we repurpose autoregressive LLMs as time series forecasters as depicted in Figure 2. Prior to this, we define time series token as the consecutive and non-overlapping segment of a single variate. It is regarded as the common token of the LLM-based forecaster, which encompasses series variations and mitigates excessively long autoregression. To focus on modeling temporal variations, our forecaster predicts each variate independently. Beyond Channel Independence [26] that implicitly captures the multivariate correlation [22] by shared parameters, AutoTimes converts timestamps into position embeddings and explicitly aligns simultaneous segment tokens, which is detailed in the next paragraph. Therefore, we simplify $\mathbf{x}_t$ as the time point of specific variate $x_t \in \mathbb{R}$. Given a single-variate time series of context length NS, the i-th segment of length S is denoted as:

$$\mathbf{s}_i = \{x_{(i-1)S+1}, \ldots, x_{iS}\} \in \mathbb{R}^S, \ i = 1, \ldots, N. \tag{2}$$

Considering the general-purpose token transition, we freeze the parameters of large language models. To realize the token-wise alignment between time series tokens and language tokens, we establish $\mathrm{SegmentEmbedding}(\cdot) : \mathbb{R}^S \mapsto \mathbb{R}^D$ that independently embeds segments into the latent space:

$$\mathbf{SE}_i = \mathrm{SegmentEmbedding}(\mathbf{s}_i), \ i = 1, \ldots, N, \tag{3}$$

where D is consistent with the dimension of the LLM.

Position embedding Timestamp, an essential covariate indicating the chronological information, is generally utilized as an extra embedding in previous deep forecasters [43, 48]. However, increasing models [10, 26, 45] have discarded the embedding and found the performance will not be greatly affected, implying the improper encoding of timestamps. In contrast, textual timestamps have been demonstrated as an enhancement in LLM4TS methods, which are always formulated into prefix-prompts [15, 21]. Nevertheless, it also leads to excessive context length, impeding LLMs from paying sufficient attention to time series tokens and inducing time-consuming feed-forwarding. Inspired by the functionality of position embedding, which incorporates information about the relative or absolute position of the tokens [37]. We adopt LLM-embedded timestamps as position embeddings to utilize temporal information and align simultaneous events (segments) from different varieties.

Technically, we formulate the starting and end timestamps of corresponding segments by the template demonstrated in Figure 3. Experimentally, we observe that the simple template without deft design can consistently boost the forecasting performance in Appendix D.5, aiding the LLM-based forecaster to comprehend the date and align different variates based on Channel Independence. Since all the previous language tokens are visible to the special ending token `<EOS>` of a sentence, we adopt the embedding of `<EOS>` as $\mathbf{TE}_i \in \mathbb{R}^D$ as the position embedding from textual timestamps:

$$\mathbf{TE}_i = \mathrm{SelectLast}\left(\mathrm{LLM}(\mathrm{TimestampTemplate}(\mathbf{s}_i))\right). \tag{4}$$

Notably, $\mathbf{TE}_i$ is pre-computed by LLMs such that runtime forwarding for language tokens is not required during training. Given that the latent space of the LLM locates both time series tokens and

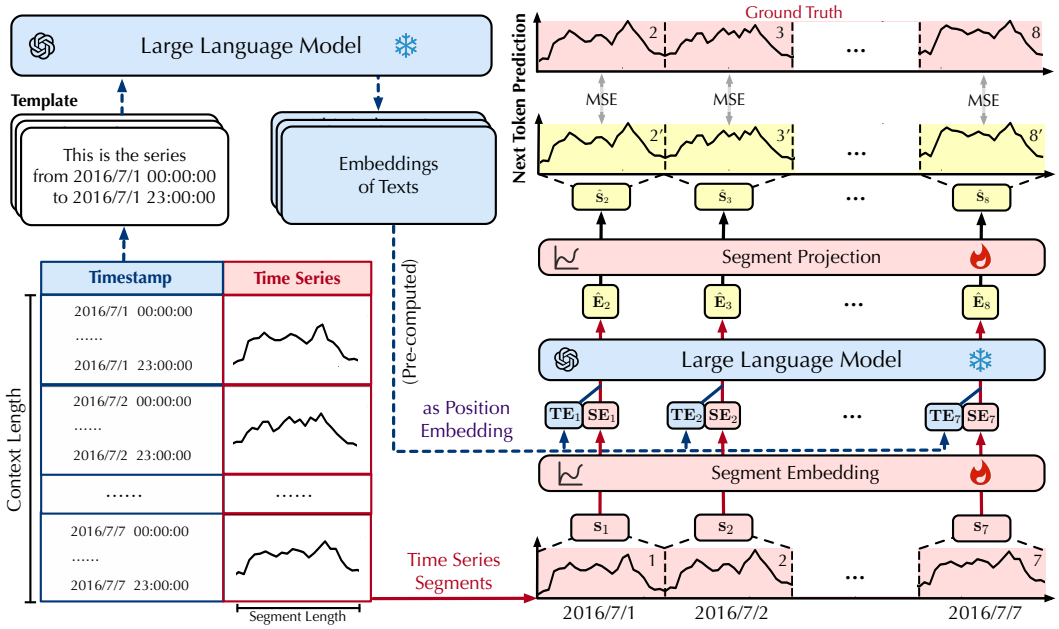

Figure 3: Overview of AutoTimes: (1) time series and corresponding timestamps are segmented; (2) textual timestamps are converted into the position embeddings by the LLM; (3) time series segments are embedded and projected by next token prediction, where intermediate layers of LLM are frozen.

language tokens, the position embedding can be integrated with the corresponding time span without increasing the context length. Concretely, the token embedding $\mathbf{E}_i \in \mathbb{R}^D$ is obtained by:

$$\mathbf{E}_i = \mathbf{SE}_i + \mathbf{TE}_i. \tag{5}$$

3.2 Next Token Prediction

As shown in Figure 3, prevalent LLMs [6, 36] are endowed with the capability of predicting the next token $\mathbf{s}_i$ based on the preceding tokens $\mathbf{s}_{<i}$. We reutilize LLMs in a fully consistent approach and generate prediction of arbitrary lengths iteratively. Given a time series of context length NS, the input series is segmented and embedded into N token embeddings $\{\mathbf{E}_1, \ldots, \mathbf{E}_N\}$. The training objective is to independently generate the next tokens $\{\hat{s}_2, \ldots, \hat{s}_{N+1}\}$. We feed the token embeddings $\mathbf{E}_i$ into the intermediate layers of the LLM, which inherently parameterize token transitions:

$$\{\hat{\mathbf{E}}_2, \ldots, \hat{\mathbf{E}}_{N+1}\} = \text{LLMLayers}(\{\mathbf{E}_1, \ldots, \mathbf{E}_N\}). \tag{6}$$

We adopt $\text{SegmentProjection}(\cdot) : \mathbb{R}^D \mapsto \mathbb{R}^S$ to independently projects embeddings to segments:

$$\hat{\mathbf{s}}_i = \text{SegmentProjection}(\hat{\mathbf{E}}_i),\ i = 2, \ldots, N+1. \tag{7}$$

Finally, each predicted segment is supervised by the token-wise ground truth to optimize the parameters of embedding and projection layers, which are simply implemented by multi-layer perceptrons:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{NS} \sum ||\mathbf{s}_i - \hat{\mathbf{s}}_i||_2^2,\ i = 2, \ldots, N. \tag{8}$$

Notably, the context length NS is decided during training, representing the maximum input length during inference. Therefore, one consequent forecaster is suitable for different input lengths like the LLM, validated in Appendix D.4. Moreover, AutoTimes can generate predictions of arbitrary lengths by iterative multi-step forecasting, proven to overcome error accumulation better than state-of-the-art forecasters in Section 4.1, since autoregressive LLMs inherently excel at multi-step generation:

$$\hat{\mathbf{s}}_i = \text{LLMForecaster}(\mathbf{s}_{<i}),\ i = 1, \ldots, \frac{F}{S}. \tag{9}$$

Instead of respectively training models on different lookback/forecast lengths, AutoTimes handles all the scenarios by one model. Surprisingly, with the consistency of autoregression, it also inherits notable generalizability and scaling behavior of LLMs, which is demonstrated in Sections 4.2 and 4.4.

3.3 In-Context Forecasting

Large language models are capable of generating expected outputs based on provided task demonstrations from downstream datasets without gradient updating, known as the in-context learning ability. The task demonstrations are generally constituted by paired questions and answers [47]. Formally, the context $\mathcal{C} = \{g(x^{(1)}, y^{(1)}), \ldots, g(x^{(m)}, y^{(m)})\}$ represents a set of demonstrations with m pairs, where $g(\cdot)$ is the template that transforms each question and answer into natural language.

In terms of time series forecasting, we propose to constitute the pair by lookback-forecast windows, which are exactly represented as successive time points from earlier historical observations. Hence, we use time series in target datasets as prompts, *extending the context for prediction beyond consecutive lookback series*. We denote the extended context as $\mathcal{C}$, which contains m time series prompts $\mathrm{tsp}^{(j)}$:

$$\mathcal{C} = \{\mathrm{tsp}^{(j)} = \mathbf{x}_{\leq t_j} | \text{ earlier historical time series}\},\ j = 1, \ldots, m,\ t_j \leq L. \tag{10}$$

During training, we first obtain an LLM-based forecaster on a source dataset and select time series prompts from the downstream target dataset based on a unified strategy. During inference, we ensure all the prompts appear before the window to be predicted, such that there is no data leakage from future information. As shown in Figure 4, we concatenate time series prompts with lookback series and feed them as the context of the forecaster, termed *in-context forecasting*:

$$f : (\mathcal{C}, \mathbf{x}_{1:L}, \mathbf{a}_{1:L+F}) \mapsto \hat{\mathbf{x}}_{L+1:L+F}. \tag{11}$$

4 Experiments

We conduct thorough evaluations of the performance of AutoTimes, including time series forecasting, zero-shot forecasting, and the proposed in-context forecasting. Additional analyses are included to evaluate the generality, scaling behavior, and adaptation cost of large language models. Detailed code implementation for reproduction is provided in our public code repository.

4.1 Time Series Forecasting

Benchmarks For long-term time series forecasting, we extensively include real-world datasets, including ETTh1, ECL, Traffic, Weather [43], and Solar-Energy [22]. For short-term forecasting, we adopt the well-acknowledged M4 competition [25]. Detailed descriptions are provided in Appendix A.

Baselines We compare AutoTimes with state-of-the-art models, including advanced LLM4TS methods: TimeLLM [15], UniTime [21], and FPT [49]; well-acknowledged deep forecasters: iTransformer [22], DLinear [45], PatchTST [26], and TimesNet [42]. For the challenging short-term forecasting, we further include competitive baselines: Koopa [23], N-HiTS [8] and N-BEATS [28]. All baselines are officially implemented or reported. We adopt LLaMA-7B [36] as our base LLM. Detailed implementations, error bars, and hyperparameter analysis are provided in Appendix B and C.

Setups For short-term forecasting, we follow the well-acknowledged TimesNet [42], which assesses the fundamental ability of forecasters in modeling temporal variations. For long-term forecasting, we establish a novel *one-for-all* benchmark: a single forecaster is trained on one dataset and subsequently utilized for all prediction lengths. We highlight that this approach evaluates the basic versatility as foundation models of time series, which aims to break the prevailing practice of extensive training across diverse real-world scenarios. To be specific, we evaluate all methods by rolling forecasting: a model is trained with predetermined input/output lengths, and the predicted values are integrated as part of the input in subsequent iterations until reaching the desired forecast length. Therefore, the key to success in this task lies in mitigating multi-step error accumulation. Still, the conventional *one-for-one* approach that trains forecasters respectively on each length is also provided in Table 12.

Results The average results are presented in Table 2- 3, with the best results in **bold** and the second best underlined. AutoTimes consistently outperforms all counterparts of short-term forecasting in Table 2, demonstrating the basic ability of LLM-based forecasters to capture diverse series variations. Further, in the one-for-all long-term scenarios, AutoTimes surpasses other LLM4TS methods and deep forecasters in 80% datasets in Table 3, outperforming previous state-of-the-art TimeLLM by

Table 2: Average short-term forecasting results on the M4 [25]. Full results are provided in Table 11.

Models		**AutoTimes**	TimeLLM	FPT	Koopa	N-HiTS	DLinear	PatchTST	TimesNet	FiLM	N-BEATS
Average	sMAPE	**11.831**	11.983	11.991	11.863	11.960	12.418	13.022	11.930	12.489	11.910
	MASE	**1.585**	1.595	1.600	1.595	1.606	1.656	1.814	1.597	1.690	1.613
	OWA	**0.850**	0.859	0.861	0.858	0.861	0.891	0.954	0.867	0.902	0.862

Table 3: Long-term forecasting results of one-for-all: we conduct rolling forecasting with a single model trained on each dataset and accomplish four desired forecast lengths in $\{96, 192, 336, 720\}$. AutoTimes adapt LLMs with the context length $C = 672$. We set the input length $L = 672$ and output length $F = 96$ in other methods. All results are averaged. Full results is provided in Table 10.

Models	**AutoTimes**		TimeLLM [15]		UniTime [21]		FPT [49]		iTrans. [22]		DLinear [45]		PatchTST [26]		TimesNet [42]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.389**	**0.422**	0.412	0.437	0.683	0.596	0.429	0.439	0.421	0.445	0.426	0.444	0.409	0.430	0.495	0.491
ECL	**0.159**	**0.253**	0.181	0.288	0.325	0.399	0.184	0.284	0.164	0.258	0.165	0.265	0.169	0.268	0.201	0.303
Weather	0.235	0.273	**0.225**	**0.266**	0.461	0.459	0.228	**0.266**	0.266	0.291	0.239	0.291	0.226	0.268	0.264	0.293
Traffic	**0.374**	**0.264**	0.410	0.303	0.584	0.367	0.461	0.326	0.384	0.274	0.423	0.298	0.391	0.275	0.602	0.322
Solar.	**0.197**	**0.242**	0.263	0.335	0.392	0.462	0.236	0.303	0.213	0.291	0.222	0.283	0.202	0.269	0.213	0.295

9.12% in average. Compared with other forecasters trained in the one-for-one scenario in Table 12, AutoTimes still achieved state-of-the-art performance in 70% of settings without respective training.

By diving into the proposed one-for-all and the traditional one-for-one benchmarks in Table 3 and 12, it is notable that prevalent deep forecasters, such as Transformer-based forecasters and DLinear, can achieve competitive and even better results under rolling forecasting. Nevertheless, the performance of non-autoregressive LLM4TS methods can degenerate a lot without respective training. Therefore, it highlights our persistent utilization of autoregression and thorough leveraging of inherent token transitions of LLMs, thereby mitigating error accumulation during multi-step rolling forecasting.

4.2 Zero-Shot Forecasting

Setups Large language models have exhibited remarkable zero-shot generalization capability [6]. To verify whether our LLM-based forecaster inherits this ability, where no training sample of the target domain is available, we assess the performance of zero-shot forecasting. Concretely, we adhere to the benchmark established by FPT [49], where the forecaster is initially trained on a source domain and subsequently evaluated on an unseen target domain. We conduct the transfer learning between the M3 and M4 competitions, both of which encompass abundant temporal variation patterns but follow different data distributions. We compare AutoTimes with deep forecasters and FPT as the only LLM4TS method, given that only FPT has exhibited zero-shot generalization in this benchmark.

Table 4: Zero-shot forecasting results in averaged SMAPE. M4 $\rightarrow$ M3 trains forecasters on the datasets of M4 and evaluates on M3, and vice versa. Detailed results are provided in Appendix D.2

Models	**AutoTimes**	FPT	DLinear	PatchTST	TimesNet	NSFormer	FEDFormer	Informer	Reformer
M4 $\rightarrow$ M3	**12.75**	13.06	14.03	13.06	14.17	15.29	13.53	15.82	13.37
M3 $\rightarrow$ M4	**13.036**	13.125	15.337	13.228	14.553	14.327	15.047	19.047	14.092

Results The comprehensive results of zero-shot forecasting are presented in Table 4. AutoTimes demonstrates superior performance compared to deep forecasters and FPT in both M4 $\rightarrow$ M3 and M3 $\rightarrow$ M4 scenarios. It is evident that LLM4TS methods generally achieve improved performance in this task due to the enhanced model capacity, leading to a 15% SMAPE reduction compared with the efficient forecaster DLinear. Despite sharing the same Transformer backbone, LLM4TS methods still outperform PatchTST due to the transferable knowledge pre-trained on large corpora of sequences. This underscores the advantage of leveraging LLMs for time series forecasting. Moreover, AutoTimes inherits general-purpose token transitions, surpassing FPT without tuning intermediate LLM layers.

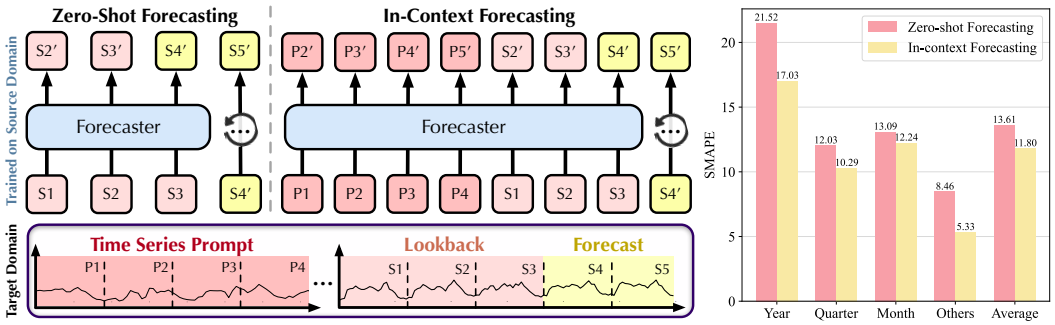

Figure 4: Demonstration of in-context forecasting and results compared with zero-shot. We uniformly select the foremost time points from the target domain as prompts and concatenate them with lookback to obtain the prediction. AutoTimes adapts LLMs on the source domain with a larger context length to place the additional time series prompt. Supplementary showcases are provided in Figure 12.

4.3 In-Context Forecasting

Setups We conduct in-context forecasting on AutoTimes, which is depicted in Figure 4. Similar to zero-shot forecasting, the task is to apply a forecaster, trained on a source dataset, to an unseen target dataset. Additionally, several task demonstrations from the target domain, referred to as time series prompts in Equation 10, are available during inference. Specifically, we concatenate these prompts with the lookback window to form the context for prediction.

We adopt the aforementioned M4 $\rightarrow$ M3 scenario. Since the samples of the M3 dataset are univariate time series with different lengths, we always predict the last F time points of each sample during inference. We set the lookback length $L = F$, and thus the length of time series prompts is $2F$. We set the number of prompts $m = 1$. Therefore, we initially train an LLM-based forecaster on the source M4 dataset with the context length of $3F$. We adopt an intuitive strategy to select the prompt: uniformly adopting the first $2F$ time points of the time series as the corresponding prompt. Supposing the lookback series starts after time t ($\geq F$), in-context forecasting is formulated as:

$$f : (\{x_{1:2F}\}, x_{t+1:t+F}, \mathbf{a}_{t+1:t+2F}) \mapsto \hat{x}_{t+F+1,t+2F}. \tag{12}$$

To prevent data leakage of future information, too short samples are discarded to prevent the overlap between the prompt and the future prediction. The implementation details of in-context forecasting are provided in Appendix D.6. We further investigate different prompt retrieval strategies. Insightful results are provided to reveal the influence of using time series prompts for interactive prediction and take-away instructions of prompt engineering in the time series modality.

Results The quantitative results of in-context forecasting are provided on the right of Figure 4. The results of zero-shot forecasting, where no downstream demonstration is available, are compared as the baseline. Benefiting from the time series prompts of the target domain, our LLM-based forecaster with the proposed in-context forecasting paradigm achieves consistent promotions on all M3 subsets and the averaged 13.3% SMAPE reduction compared with zero-shot forecasting. In contrast to previous LLM4TS methods that rely on deft language prompts, LLMs adopted by AutoTimes can be instructed by time series itself with our intuitive prompting engineering. From the perspective of the forecasting paradigm, we extend the prediction context beyond the lookback window. To inherit token transitions of language models parameterized by intermediate layers, AutoTimes takes a crucial step by establishing a mapping between time series segments and the latent space of language tokens, which is however absent in non-autoregressive LLM4TS methods. Therefore, ensuring autoregression consistency enhances the effective utilization of LLMs as foundation models.

4.4 Method Analysis

Generality Previous LLM4TS [15, 49] methods focus on applying their approach to specific LLMs. We demonstrate that AutoTimes is compatible with any decoder-only LLMs. By extensively training LLM-based forecasters by AutoTimes based on prevalent LLMs, including GPT-2 [31], OPT [46], and LLaMA [36], we present the results in Table 5, highlighting the generality of AutoTimes.

Table 5: Averaged results of alternative language models. Full results are provided in Table 18.

LLM	GPT-2 (124M)		OPT-350M		OPT-1.3B		OPT-2.7B		OPT-6.7B		LLaMA-7B	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	0.173	0.266	0.168	0.263	0.164	0.258	0.164	0.258	0.162	0.256	0.159	0.253
ETTh1	0.397	0.425	0.401	0.429	0.396	0.424	0.394	0.424	0.394	0.424	0.389	0.423
Traffic	0.406	0.276	0.405	0.277	0.397	0.271	0.394	0.269	0.393	0.270	0.374	0.264
Weather	0.242	0.278	0.240	0.275	0.240	0.276	0.243	0.277	0.247	0.282	0.235	0.273

Scaling behavior Scalability is an essential characteristic that emerges from small models to large foundation models. By investigating the results presented in Table 5, we observe that the prediction accuracy of the forecaster generally improves with the increase in LLM parameters. This scaling behavior of LLM-based forecasters introduces a trade-off between performance and adaptation cost. To provide a comprehensive assessment, we evaluate each adapted forecaster from three perspectives: performance, training speed, and parameters, as presented in Figure 5. We observe that the largest LLaMA-7B consistently delivers optimal forecasting performance. As a relatively small language model, OPT-1.3B exhibits good parameter efficiency as an out-of-the-box forecaster.

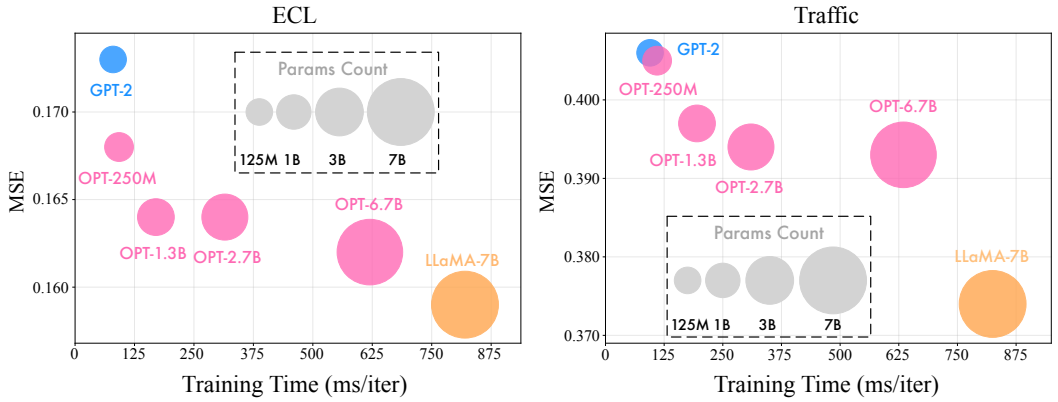

Figure 5: Efficiency comparison of alternative LLMs, evaluated by the same configuration of Table 5.

Adaptation cost To mitigate the substantial cost of adapting large language models, AutoTimes introduces minimal parameters with all intermediate layers of LLM frozen. Additionally, we seamlessly integrate the language tokens (e.g. textual timestamps) without excessive context length and runtime overhead for training, thereby significantly reducing the adaptation cost. Figure 6 presents a comprehensive efficiency analysis with advanced LLM4TS methods: FPT is applicable on GPT-2 and TimeLLM is applicable on LLaMA-7B. Not only does AutoTime achieve better results in Table 3, but its training and reasoning time is also greatly reduced, bringing over 5× speedup on average. In terms of parameter efficiency, AutoTimes focuses on establishing the embedding for time series segments, which is simply implemented by the MLP ($0.79M$) account for 0.1% parameters of the LLM ($7B$). Therefore, the results affirm the effectiveness of reutilizing the inherent token transition.

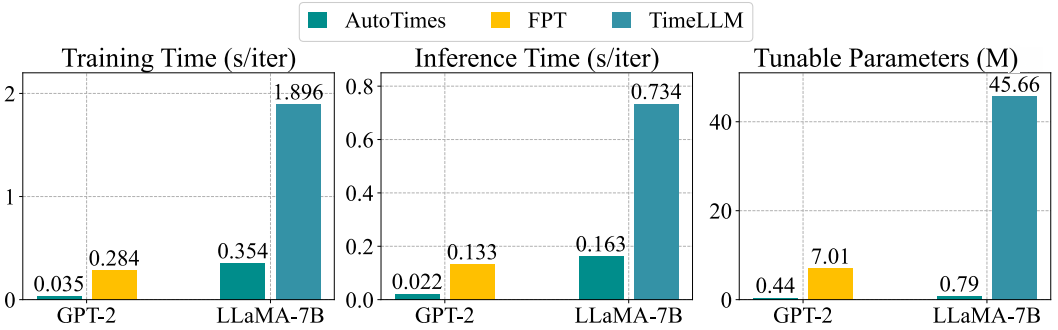

Figure 6: Comparison of AutoTimes and other LLM4TS methods in terms of training/inference time and tunable parameters with the same batch size (224) on the ETTh1 dataset.

Ablation study Recent research has raised doubts about the validity of previous LLM4TS methods [35], which predominantly adopt non-autoregression, that is, treating the LLM as a BERT-style pre-trained backbone and utilize a globally flattening projector on all lookback tokens. Here, we provide a thorough ablation study to examine our proposed AutoTimes in Table 6. The results underscore that our method maintains the consistency of the decoder-only architecture and autoregressive inference, effectively leveraging LLMs and addressing the concerns regarding performance improvement and adaptation cost. Further, we provide a comparison by substituting our token-wise segment projection (consistent with LLMs) with the flatten linear head [26] (common in non-autoregressive forecasters). Results of Table 21 in the Appendix reveal that the performance of non-autoregressive generation is consistently inferior to that of our autoregressive AutoTimes approach.

Table 6: We follow the protocol of LLM4TS ablation studies [35] to verify whether the LLM is truly useful in our AutoTimes: (1) *w/o LLM* replaces the language model entirely and passing input tokens directly to the last layer; (2) *LLM2Attn* replaces the language model with a single multi-head attention layer; (3) *LLM2Trsf* replaces the language model with a single transformer block.

Dataset	ETTh1								ECL							
Type	**AutoTimes**		w/o LLM		LLM2Attn		LLM2Trsf		**AutoTimes**		w/o LLM		LLM2Attn		LLM2Trsf	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Pred-96	**0.360**	**0.400**	0.365	0.399	0.383	0.404	0.377	0.401	**0.129**	**0.225**	0.171	0.263	0.156	0.255	0.162	0.263
Pred-192	**0.388**	**0.419**	0.405	0.425	0.414	0.422	0.406	0.420	**0.147**	**0.241**	0.192	0.282	0.178	0.276	0.189	0.287
Pred-336	**0.401**	**0.429**	0.429	0.441	0.431	0.432	0.421	0.431	**0.162**	**0.258**	0.216	0.304	0.198	0.295	0.216	0.309
Pred-720	**0.406**	**0.440**	0.450	0.468	0.456	0.454	0.449	0.452	**0.199**	**0.288**	0.264	0.342	0.230	0.320	0.258	0.340

LoRA adaptation By incorporating low-rank adaptation technique [14] on the intermediate LLM layers, the token transition of the large language model can be further fine-tuned to align the future extrapolation of time series. Table 7 provides the performance comparing the incorporation of LoRA, which consistently improves the performance of the LLM-based forecaster adapted by AutoTimes.

Table 7: Full long-term forecasting results of AutoTimes and AutoTimes equipped with LoRA [14].

Dataset		ETTh1		ECL		Weather		Traffic		Solar-Energy	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Pred-96	AutoTimes	0.360	**0.397**	0.140	0.236	0.158	0.208	0.369	0.257	0.179	0.220
	+ LoRA	**0.357**	**0.397**	**0.130**	**0.225**	**0.151**	**0.201**	**0.360**	**0.256**	**0.176**	**0.219**
Pred-192	AutoTimes	**0.391**	**0.419**	0.159	0.253	0.207	0.254	0.394	0.268	0.198	0.236
	+ LoRA	**0.391**	0.420	**0.149**	**0.242**	**0.197**	**0.244**	**0.383**	**0.267**	**0.195**	**0.235**
Pred-336	AutoTimes	**0.408**	**0.432**	0.177	0.270	0.262	0.298	0.413	0.278	0.213	0.252
	+ LoRA	0.409	0.433	**0.164**	**0.259**	**0.251**	**0.287**	**0.401**	**0.277**	**0.208**	**0.249**
Pred-720	AutoTimes	0.429	0.452	0.216	0.303	0.342	0.353	0.449	**0.299**	0.239	0.277
	+ LoRA	**0.426**	**0.451**	**0.202**	**0.293**	**0.326**	**0.339**	**0.440**	0.300	**0.225**	**0.268**

5 Conclusion

This paper aims to develop foundation models for time series forecasting. We utilize off-the-shelf LLMs as autoregressive forecasters by transferring the general-purpose and multi-step generation ability. Different from prior methods, we notice prevalent non-autoregressive LLM4TS methods may contradict the decoder-only structure and lead to insufficient utilization of LLMs. Experimentally, the proposed method achieves state-of-the-art performance with remarkable model efficiency. Further analysis reveals that our forecaster effectively inherits advanced capabilities such as zero-shot and in-context forecasting, and is able to utilize both instructive times series and timestamps. In the future, we will further incorporate advanced low-rank adaptation and utilize booming language backbones.

Acknowledgments

This work was supported by the Ministry of Industry and Information Technology of China.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[2] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

[3] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[4] George Box. Box and jenkins: time series analysis, forecasting and control. In *A Very British Affair: Six Britons and the Development of Time Series Analysis During the 20th Century*, pages 161–215. Springer, 2013.

[5] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[6] 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] 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. *arXiv preprint arXiv:2310.04948*, 2023.

[8] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6989–6997, 2023.

[9] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*, 2022.

[10] 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.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[12] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*, volume 38. OUP Oxford, 2012.

[13] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. *arXiv preprint arXiv:2310.07820*, 2023.

[14] 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.

[15] 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. *arXiv preprint arXiv:2310.01728*, 2023.

[16] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[19] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022.

[20] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[21] Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. Unitime: A language-empowered unified model for cross-domain time series forecasting. *arXiv preprint arXiv:2310.09751*, 2023.

[22] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

[23] Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with koopman predictors. *arXiv preprint arXiv:2305.18803*, 2023.

[24] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Transformers for time series analysis at scale. *arXiv preprint arXiv:2402.02368*, 2024.

[25] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020.

[26] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[27] R OpenAI. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2:13, 2023.

[28] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[33] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[34] Chenxi Sun, Yaliang Li, Hongyan Li, and Shenda Hong. Test: Text prototype aligned embedding to activate llm's ability for time series. *arXiv preprint arXiv:2308.08241*, 2023.

[35] Mingtian Tan, Mike A Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? *arXiv preprint arXiv:2406.16964*, 2024.

[36] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[37] 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.

[38] Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What language model architecture and pretraining objective works best for zero-shot generalization? In *International Conference on Machine Learning*, pages 22964–22984. PMLR, 2022.

[39] Peter R Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[40] Gerald Woo, Chenghao Liu, Akshat Kumar, and Doyen Sahoo. Pushing the limits of pre-training for time series forecasting in the cloudops domain. *arXiv preprint arXiv:2310.05063*, 2023.

[41] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592*, 2024.

[42] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

[43] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

[44] Hao Xue and Flora D Salim. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[45] 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.

[46] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[47] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[48] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai 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.

[49] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained lm. *arXiv preprint arXiv:2302.11939*, 2023.

[50] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

A Dataset Descriptions

We conduct experiments to evaluate the performance of the proposed AutoTimes on seven real-world datasets spanning diverse domains: (1) ETTh1 [48] spans from July 2016 to July 2018 and consists of seven factors related to electricity transformers. (2) Weather [43] encompasses 21 meteorological factors collected every 10 minutes in 2020 from the Weather Station of the Max Planck Biogeochemistry Institute. (3) ECL [43] captures hourly electricity consumption data from 321 clients. (4) Traffic [43] gathers hourly road occupancy rates from 862 sensors on San Francisco Bay area freeways, covering the period from January 2015 to December 2016. (5) Solar-Energy [18] records solar power production from 137 PV plants in 2006, sampled every 10 minutes. (6) M4 is a competition dataset encompassing various time series across different frequencies and domains such as business and economics. (7) M3, albeit smaller than M4, also contains diverse time series from various domains.

We follow the same data processing and train-validation-test set split protocol used in TimesNet [43], where the train, validation, and test datasets are strictly divided according to chronological order to ensure no data leakage. As for long-term forecasting settings, we fix the context length of AutoTimes and the lookback length of other compared methods as 672 in ETT, ECL, Traffic, Weather, and Solar-Energy, and the forecast length varies in {96, 192, 336, 720}. For the short-term forecasting on M4 and M3 datasets, the input length is generally set to twice the output length according to the official implementation of TimesNet. The details are provided in Table 8.

Table 8: Detailed dataset descriptions. *Dim* denotes the variate number. *Dataset Size* denotes the total number of time points in (Train, Validation, Test) splits respectively. *Forecast Length* denotes the future time points to be predicted. *Frequency* denotes the sampling interval of time points.

Dataset	Dim	Forecast Length	Dataset Size	Frequency	Information
ETTh1	7	{96, 192, 336, 720}	(8545, 2881, 2881)	Hourly	Electricity
Weather	21	{96, 192, 336, 720}	(36792, 5271, 10540)	10min	Weather
ECL	321	{96, 192, 336, 720}	(18317, 2633, 5261)	Hourly	Electricity
Traffic	862	{96, 192, 336, 720}	(12185, 1757, 3509)	Hourly	Transportation
Solar-Energy	137	{96, 192, 336, 720}	(36601, 5161, 10417)	10min	Energy
M4-Yearly	1	6	(23000, 0, 23000)	Yearly	Demographic
M4-Quarterly	1	8	(24000, 0, 24000)	Quarterly	Finance
M4-Monthly	1	18	(48000, 0, 48000)	Monthly	Industry
M4-Weekly	1	13	(359, 0, 359)	Weekly	Macro
M4-Daily	1	14	(4227, 0, 4227)	Daily	Micro
M4-Hourly	1	48	(414, 0, 414)	Hourly	Other
M3-Yearly	1	6	(645, 0, 645)	Yearly	Demographic
M3-Quarterly	1	8	(756, 0, 756)	Quarterly	Finance
M3-Monthly	1	18	(1428, 0, 1428)	Monthly	Industry
M3-Others	1	8	(174, 0, 174)	Weekly	Macro

B Implementation Details

AutoTimes processes timestamps in textual form rather than numerical encoding, potentially enabling to handle other textual data such as news or logs. We utilize LLM to obtain embedding for the special token `<EOS>` to capture embedding for the entire sentence. Pseudo-code for this process is depicted in Algorithm 1. It is worth noting that in the context of multivariate time series forecasting, timestamps are shared across variates. Thus, timestamps can implicitly express relationships between variates even with channel independence. Further, assuming there are C variates since the number of timestamps is $\frac{1}{C}$ of the total time point count, these embeddings can be efficiently pre-computed by large language models.

After obtaining embedding for the timestamps, we repurpose LLM for time series forecasting using Algorithm 2. At this stage, only the parameters of $\mathrm{SegmentEmbedding}$ and $\mathrm{SegmentProjection}$ are updated, while the

parameters of LLMs remain entirely frozen. During inference, AutoTimes utilizes the last token generated as its prediction and then employs this output to create subsequent predictions autoregressively. This approach enables AutoTimes to predict sequences of variable lengths with just one model dynamically. Such capability is crucial in real-world application scenarios. The pseudo-code in Algorithm 3-4 illustrates this process.

All the experiments are conducted using PyTorch [29] on NVIDIA A100 GPUs. We employ Adam [17] with an initial learning rate in $\{10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$ and MSE loss for model optimization. We adopt Channel Independence [26] for multivariate time series and utilize our position embeddings of timestamps to explicitly align them. The batch size is chosen from $\{256, 1024, 2048\}$, and we set the number of training epochs as 10. As for $\mathrm{SegmentEmbedding}$ and $\mathrm{SegmentProjection}$, we implement them by either a linear layer or MLP. Results of deep forecaster are based on the benchmark provided by the TimesNet [43] repository, which is fairly built on the same configurations provided by the original paper. LLM4TS methods [15, 21, 49] are implemented by their official and open-source repository. Unless otherwise specified, we use LLaMA-7B [36] as the default base LLM. We also present the standard deviation of AutoTimes forecasting performance with three random seeds in Table 9, demonstrating that the performance of AutoTimes is stable.

Algorithm 1 AutoTimes - Generate Text Embedding

Require: Input time series $\mathbf{x}_{(i-1)S+1:iS}$ ▷ i-th token of length S

$\mathbf{s}_i = \{x_{(i-1)S+1}, \ldots, x_{iS}\}$ ▷ Model dimension of the LLM D

$\mathbf{TE}_i = \mathrm{SelectLast}\left(\mathrm{LLM}(\mathrm{TextualDuration}(\mathbf{s}_i))\right)$ ▷ $\mathbf{TE}_i \in \mathbb{R}^D$

Return $\mathbf{TE}_i$ ▷ Return textual embedding of i-th token

Algorithm 2 AutoTimes - Repurpose LLM

Require: Input time series $\{x_1, \ldots, x_{(N+1)\times S}\}$; text embeddings $\{\mathbf{TE}_1, \ldots, \mathbf{TE}_N\}$ ▷ Token number N

for i **in** $\{1, \ldots, N\}$**:**

  $\mathbf{s}_i = \{x_{(i-1)S+1}, \ldots, x_{iS}\}$ ▷ $\mathbf{s}_i \in \mathbb{R}^S$

  $\mathbf{SE}_i = \mathrm{SegmentEmbedding}(\mathbf{s}_i)$ ▷ $\mathbf{SE}_i \in \mathbb{R}^D$

  $\mathbf{E}_i = \mathbf{SE}_i + \mathbf{TE}_i$ ▷ $\mathbf{E}_i \in \mathbb{R}^D$

$\{\hat{\mathbf{E}}_2, \ldots, \hat{\mathbf{E}}_{N+1}\} = \mathrm{LLMLayers}(\{\mathbf{E}_1, \ldots, \mathbf{E}_N\})$ ▷ $\hat{\mathbf{E}}_i \in \mathbb{R}^D$

for i **in** $\{2, \ldots, N+1\}$**:**

  $\hat{\mathbf{s}}_i = \mathrm{SegmentProjection}(\hat{\mathbf{E}}_i)$ ▷ $\hat{\mathbf{s}}_i \in \mathbb{R}^S$

$\mathbf{s}_{N+1} = \{x_{NS+1}, \ldots, x_{NS+S}\}$

$\mathcal{L}_{\text{MSE}} = \frac{1}{NS} \sum ||\mathbf{s}_i - \hat{\mathbf{s}}_i||_2^2,\ i \in \{2, \ldots, N+1\}$ ▷ Objective of the next token prediction

Algorithm 3 AutoTimes - LLM Forecasting

Require: Input time series $\{x_1, \ldots, x_{N_1 \times S}\}$; text embeddings $\{\mathbf{TE}_1, \ldots, \mathbf{TE}_{N_1}\}$ ▷ Lookback token number $N_1 \leq N$

for i **in** $\{1, \ldots, N_1\}$**:**

  $\mathbf{s}_i = \{x_{(i-1)S+1}, \ldots, x_{iS}\}$ ▷ $\mathbf{s}_i \in \mathbb{R}^S$

  $\mathbf{SE}_i = \mathrm{SegmentEmbedding}(\mathbf{s}_i)$ ▷ $\mathbf{SE}_i \in \mathbb{R}^D$

  $\mathbf{E}_i = \mathbf{SE}_i + \mathbf{TE}_i$ ▷ $\mathbf{E}_i \in \mathbb{R}^D$

$\{\hat{\mathbf{E}}_2, \ldots, \hat{\mathbf{E}}_{N_1+1}\} = \mathrm{LLMLayers}(\{\mathbf{E}_1, \ldots, \mathbf{E}_{N_1}\})$ ▷ $\hat{\mathbf{E}}_i \in \mathbb{R}^D$

for i **in** $\{2, \ldots, N+1\}$**:**

  $\hat{\mathbf{s}}_i = \mathrm{SegmentProjection}(\hat{\mathbf{E}}_i)$ ▷ $\hat{\mathbf{s}}_i \in \mathbb{R}^S$

Return $\hat{\mathbf{s}}_{N+1}$ ▷ Return last token $\hat{\mathbf{s}}_{N+1} \in \mathbb{R}^S$ as the prediction

Algorithm 4 AutoTimes - Autoregressive Generation

Require: Input time series $\{x_1, \ldots, x_{N_1 \times S}\}$; textual embeddings $\{\mathbf{TE}_1, \ldots, \mathbf{TE}_{N_2}\}$ ▷ Forecast token number $N_2 - N_1$

$\mathbf{x} = \{x_1, \ldots, x_{N_1 \times S}\}$

prediction = {}

for i **in** $\{1, \ldots, N_2 - N_1\}$**:**

  $\hat{\mathbf{y}} = \text{LLMForecaster}(\mathbf{x}, \mathbf{TE}_{:N_1+i-1})$ ▷ Details in Algorithm 3

  $\mathbf{x} \leftarrow \{\mathbf{x}, \hat{\mathbf{y}}\}$ ▷ Concatenate for the input for next iteration

  prediction $\leftarrow \{\text{prediction}, \hat{\mathbf{y}}\}$ ▷ Record prediction results

Return prediction ▷ Return result $\in \mathbb{R}^{(N_2-N_1)\times S}$

Table 9: Performance and standard deviations of AutoTimes. Results come from three random seeds.

Dataset	ETTh1		ECL		Weather	
Horizon	MSE	MAE	MSE	MAE	MSE	MAE
96	0.360±0.002	0.400±0.002	0.129±0.001	0.225±0.001	0.153±0.000	0.203±0.001
192	0.388±0.002	0.419±0.001	0.147±0.002	0.241±0.002	0.201±0.001	0.250±0.001
336	0.401±0.003	0.429±0.002	0.162±0.002	0.258±0.003	0.256±0.002	0.293±0.001
720	0.406±0.004	0.440±0.002	0.199±0.004	0.288±0.002	0.331±0.002	0.345±0.001
Dataset	Traffic		Solar-Energy			
Horizon	MSE	MAE	MSE	MAE		
96	0.343±0.001	0.248±0.001	0.171±0.001	0.221±0.001		
192	0.362±0.001	0.257±0.001	0.190±0.001	0.236±0.002		
336	0.379±0.002	0.266±0.001	0.203±0.002	0.248±0.002		
720	0.413±0.003	0.284±0.002	0.222±0.002	0.262±0.003		

C Hyperparameter Sensitivity

SegmentEmbedding and SegmentProjection are uniformly implemented by MLP. The number of layers is fixed as 2 and the hidden dimension is selected from $\{256, 512, 1024\}$ according to the validation loss. The segment length is set as $S = 96$ in multivariate datasets and is set as the prediction length $S = F$ in M3 and M4. We verify the robustness of AutoTimes of hyperparameters as follows: the layer number and hidden dimension of SegmentEmbedding and SegmentProjection, context length, and segment length. We observe that AutoTimes is insensitive to the configurations of embedding and projection layers. Besides, performance can be improved by increasing the context length. For long prediction lengths, a larger segment length is favored.

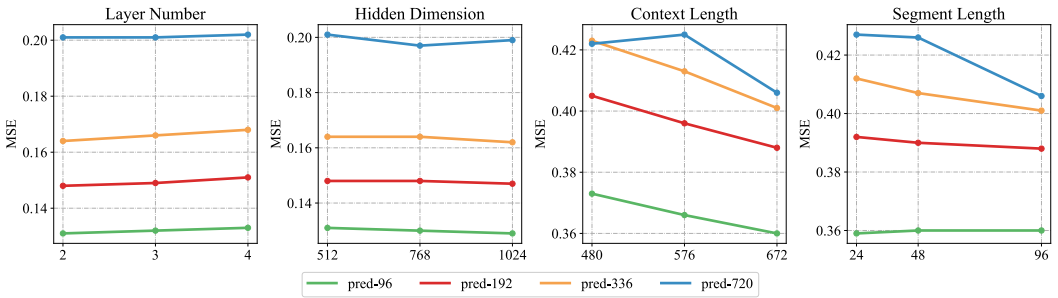

Figure 7: Hyperparameter sensitivity of AutoTimes. Each curve presents a specific forecast length.

D Supplementary Results

D.1 Time Series Forecasting

We compare the performance of AutoTimes with state-of-the-art LLM4TS methods and well-acknowledged deep forecasters. Table 11 shows detailed short-term forecast results on M4. Table 10 presents results of the one-for-all forecasting benchmark across ETTh1, ECL, Traffic, Weather, and Solar-Energy datasets. We evaluate all methods by rolling forecasting: each model is trained with input length 672 and output length 96, and the predicted values are integrated as part of the input in the next iteration until reaching the desired forecast horizon.

Furthermore, the traditional one-for-one approach, where forecasters are trained individually for each prediction length, is also presented in Table 12. The results are reproduced using their corresponding official code. For the sake of rigor, we also provide our reproduced results with the officially reported results in Table 13.

Additionally, we evaluate AutoTimes along with other baseline models on recent benchmarks [24]. Results are presented in Table 15. We also look forward to evaluating on more diverse benchmarks in the future.

Table 10: Full long-term forecasting results of one-for-all: we conduct rolling forecasting with a single model trained on each dataset and accomplish four desired forecast lengths in $\{96, 192, 336, 720\}$. AutoTimes adapt LLMs with the context length $C = 672$. We set the input length $L = 672$ and output length $F = 96$ in other methods, which are all implemented by their official code.

Method		**AutoTimes**		TimeLLM [15]		UniTime [21]		FPT [49]		iTrans. [22]		DLinear [45]		PatchTST [26]		TimesNet [42]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.360**	**0.400**	0.380	0.412	0.386	0.409	0.377	0.404	0.387	0.418	0.369	**0.400**	0.374	0.401	0.452	0.463
	192	**0.388**	**0.419**	0.408	0.431	0.639	0.577	0.412	0.426	0.416	0.437	0.405	0.422	0.405	0.422	0.474	0.477
	336	**0.401**	**0.429**	0.425	0.443	0.814	0.680	0.440	0.444	0.434	0.450	0.435	0.445	0.423	0.435	0.493	0.489
	720	**0.406**	**0.440**	0.434	0.463	0.891	0.719	0.488	0.483	0.447	0.473	0.493	0.508	0.434	0.460	0.560	0.534
	Avg	**0.389**	**0.422**	0.412	0.437	0.683	0.596	0.429	0.439	0.421	0.445	0.426	0.444	0.409	0.430	0.495	0.491
ECL	96	**0.129**	**0.225**	0.137	0.244	0.171	0.266	0.137	0.236	0.133	0.229	0.138	0.238	0.132	0.232	0.184	0.288
	192	**0.147**	**0.241**	0.158	0.266	0.293	0.378	0.158	0.258	0.151	0.245	0.152	0.251	0.151	0.250	0.192	0.295
	336	**0.162**	**0.258**	0.183	0.292	0.379	0.448	0.181	0.285	0.168	0.262	0.167	0.268	0.171	0.272	0.200	0.303
	720	**0.199**	**0.288**	0.247	0.348	0.455	0.502	0.258	0.355	0.205	0.294	0.203	0.302	0.222	0.318	0.228	0.325
	Avg	**0.159**	**0.253**	0.181	0.288	0.325	0.399	0.184	0.284	0.164	0.258	0.165	0.265	0.169	0.268	0.201	0.303
Weather	96	0.153	0.203	**0.149**	**0.200**	0.180	0.223	0.154	0.205	0.174	0.225	0.169	0.229	**0.149**	0.202	0.169	0.228
	192	0.201	0.250	**0.193**	**0.243**	0.450	0.451	0.196	0.243	0.227	0.268	0.211	0.268	0.194	0.245	0.222	0.269
	336	0.256	0.293	**0.243**	**0.284**	0.594	0.570	0.244	0.282	0.290	0.309	0.258	0.306	0.244	0.285	0.290	0.310
	720	0.331	0.345	**0.315**	**0.336**	0.618	0.593	0.318	0.335	0.374	0.360	0.320	0.362	0.317	0.338	0.376	0.364
	Avg	0.235	0.273	**0.225**	**0.266**	0.461	0.459	0.228	0.266	0.266	0.291	0.239	0.291	0.226	0.268	0.264	0.293
Traffic	96	**0.343**	**0.248**	0.376	0.280	0.438	0.291	0.395	0.283	0.353	0.259	0.399	0.285	0.359	0.255	0.593	0.315
	192	**0.362**	**0.257**	0.397	0.294	0.538	0.353	0.425	0.302	0.373	0.267	0.409	0.290	0.377	0.265	0.596	0.317
	336	**0.379**	**0.266**	0.420	0.311	0.621	0.389	0.463	0.328	0.386	0.275	0.422	0.297	0.393	0.276	0.600	0.319
	720	**0.413**	**0.284**	0.448	0.326	0.737	0.435	0.560	0.392	0.425	0.296	0.461	0.319	0.436	0.305	0.619	0.335
	Avg	**0.374**	**0.264**	0.410	0.303	0.584	0.367	0.461	0.326	0.384	0.274	0.423	0.298	0.391	0.275	0.602	0.322
Solar.	96	0.171	**0.221**	0.224	0.289	0.223	0.274	0.196	0.261	0.183	0.265	0.193	0.258	**0.168**	0.237	0.180	0.272
	192	0.190	**0.236**	0.248	0.315	0.373	0.431	0.219	0.284	0.205	0.283	0.214	0.274	**0.189**	0.257	0.199	0.286
	336	**0.203**	**0.248**	0.269	0.338	0.445	0.536	0.245	0.311	0.224	0.299	0.233	0.291	0.212	0.277	0.220	0.301
	720	**0.222**	**0.262**	0.310	0.396	0.526	0.608	0.285	0.354	0.239	0.316	0.246	0.307	0.240	0.305	0.251	0.321
	Avg	**0.197**	**0.242**	0.263	0.335	0.392	0.462	0.236	0.303	0.213	0.291	0.222	0.283	0.202	0.269	0.213	0.295

Table 11: Full results of short-term forecasting. We follow the same protocol of TimesNet [42].

Method		**AutoTimes**	TimeLLM	FPT	Koopa	N-HiTS	N-BEATS	PatchTST	TimesNet	DLinear	FiLM
Yearly	SMAPE	**13.319**	13.419	13.531	13.352	13.371	13.866	13.517	13.394	14.012	13.466
	MASE	**2.993**	3.005	3.015	2.997	3.025	3.006	3.031	3.004	3.071	3.059
	OWA	**0.784**	0.789	0.793	0.786	0.790	0.802	0.795	0.787	0.815	0.797
Quarterly	SMAPE	10.101	10.110	10.177	10.159	10.454	10.689	10.847	10.101	10.758	**10.074**
	MASE	1.182	1.178	1.194	1.189	1.219	1.294	1.315	1.183	1.306	**1.163**
	OWA	0.890	0.889	0.897	0.895	0.919	0.957	0.972	0.890	0.905	**0.881**
Monthly	SMAPE	**12.710**	12.980	12.894	12.730	12.794	13.372	14.584	12.866	13.377	12.801
	MASE	**0.934**	0.963	0.956	0.953	0.960	1.014	1.169	0.964	1.021	0.955
	OWA	**0.880**	0.903	0.897	0.901	0.895	0.940	1.055	0.894	0.944	0.893
Others	SMAPE	4.843	4.795	4.940	4.861	**4.696**	4.894	6.184	4.982	5.259	5.008
	MASE	3.277	3.178	3.228	**3.124**	3.130	3.358	4.818	3.323	3.608	3.443
	OWA	1.026	1.006	1.029	1.004	**0.988**	1.044	1.140	1.048	1.122	1.070
Average	SMAPE	**11.831**	11.983	11.991	11.863	11.960	12.418	13.022	11.930	12.489	11.910
	MASE	**1.585**	1.595	1.600	1.595	1.606	1.656	1.814	1.597	1.690	1.613
	OWA	**0.850**	0.859	0.861	0.858	0.861	0.891	0.954	0.867	0.902	0.862

Table 12: Long-term forecasting results of one-for-one: AutoTimes trains one LLM-based forecaster to handle all prediction lengths by autoregression, whereas other models are trained respectively on each prediction length. AutoTimes adapts LLMs with the context length $C = 672$. The lookback length is set as $L = 672$ in others. All results are averaged. Full results is provided in Table 14.

Models	**AutoTimes**		TimeLLM [15]		UniTime [21]		FPT [49]		iTrans. [22]		DLinear [45]		PatchTST [26]		TimesNet [42]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.389**	**0.422**	0.409	0.432	0.438	0.445	0.426	0.438	0.438	0.450	0.423	0.437	0.413	0.431	0.458	0.450
ECL	**0.159**	**0.253**	0.170	0.275	0.194	0.287	0.167	0.264	0.161	0.256	0.177	0.274	**0.159**	**0.253**	0.192	0.295
Weather	0.235	0.273	**0.227**	0.266	0.260	0.283	0.231	0.269	0.238	0.272	0.240	0.300	0.226	**0.264**	0.259	0.287
Traffic	**0.374**	**0.264**	0.402	0.294	0.460	0.301	0.416	0.295	0.379	0.272	0.434	0.295	0.391	**0.264**	0.620	0.336
Solar.	0.197	**0.242**	0.234	0.293	0.254	0.291	0.229	0.296	0.202	0.269	0.217	0.278	**0.189**	0.257	0.200	0.268

Table 13: Results of LLM4TS methods from the original paper and our reproduction by official code.

Models	**AutoTimes**		TimeLLM* [15]		TimeLLM [15]		FPT* [49]		FPT [49]		UniTime* [21]		UniTime [21]	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.389**	**0.422**	0.408	0.423	0.409	0.432	0.427	0.426	0.426	0.438	0.442	0.448	0.438	0.445
ECL	**0.159**	**0.253**	**0.159**	**0.253**	0.170	0.275	0.167	0.263	0.167	0.264	0.216	0.305	0.194	0.287
Weather	0.235	0.273	**0.225**	**0.257**	**0.227**	0.266	0.237	0.270	0.231	0.269	0.253	0.276	0.260	0.283
Traffic	**0.374**	**0.264**	0.388	**0.264**	0.402	0.294	0.414	0.294	0.416	0.295	-	-	0.460	0.301
Solar.	**0.197**	**0.242**	-	-	0.234	0.293	-	-	0.229	0.296	-	-	0.254	0.291

[1] Methods with * means results from the original paper; without * means the reproduction.
[2] "-" indicates that results are not reported in the original paper.

Table 14: Full long-term forecasting results of one-for-one: AutoTimes trains one LLM-based forecaster to handle all prediction lengths by autoregression, whereas other models are trained respectively on each prediction length. AutoTimes adapt LLMs with the context length $C = 672$. The lookback length is set as $L = 672$ in others, which are all implemented by their official code.

Method		One-for-all		Trained respectively on specific lookback/prediction length													
		AutoTimes		TimeLLM [15]		UniTime [21]		FPT [49]		iTrans. [22]		DLinear [45]		PatchTST [26]		TimesNet [42]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.360**	0.400	0.380	0.412	0.386	0.409	0.377	0.404	0.386	0.405	0.375	**0.399**	0.370	**0.399**	0.384	0.402
	192	**0.388**	0.419	0.405	0.422	0.428	0.436	0.413	0.424	0.422	0.439	0.405	**0.416**	0.413	0.421	0.557	0.436
	336	**0.401**	**0.429**	0.422	0.433	0.464	0.456	0.436	0.444	0.444	0.457	0.439	0.443	0.422	0.436	0.491	0.469
	720	**0.406**	**0.440**	0.430	0.459	0.473	0.479	0.477	0.481	0.500	0.498	0.472	0.490	0.447	0.466	0.521	0.500
	Avg	**0.389**	**0.422**	0.409	0.432	0.438	0.445	0.426	0.438	0.438	0.450	0.423	0.437	0.413	0.431	0.458	0.450
ECL	96	**0.129**	0.225	0.137	0.244	0.171	0.266	0.137	0.236	0.132	0.227	0.153	0.237	**0.129**	**0.222**	0.168	0.272
	192	**0.147**	0.241	0.162	0.271	0.178	0.274	0.154	0.251	0.153	0.249	0.152	0.249	**0.147**	**0.240**	0.184	0.289
	336	**0.162**	**0.258**	0.175	0.279	0.194	0.289	0.169	0.267	0.167	0.262	0.169	0.267	0.163	0.259	0.198	0.300
	720	0.199	0.288	0.207	0.306	0.232	0.319	0.207	0.300	**0.192**	**0.285**	0.233	0.344	0.197	0.290	0.220	0.320
	Avg	**0.159**	**0.253**	0.170	0.275	0.194	0.287	0.167	0.264	0.161	0.256	0.177	0.274	**0.159**	**0.253**	0.192	0.295
Weather	96	0.153	0.203	**0.149**	0.200	0.180	0.223	0.154	0.205	0.163	0.211	0.152	0.237	**0.149**	**0.198**	0.172	0.220
	192	0.201	0.250	0.195	0.243	0.226	0.261	0.196	0.245	0.205	0.250	0.220	0.282	**0.194**	**0.241**	0.219	0.261
	336	0.256	0.293	**0.245**	**0.282**	0.280	0.300	0.254	0.290	0.254	0.289	0.265	0.319	**0.245**	**0.282**	0.280	0.306
	720	0.331	0.345	0.318	0.338	0.355	0.348	0.321	0.337	0.329	0.340	0.323	0.362	**0.314**	**0.334**	0.365	0.359
	Avg	0.235	0.273	0.227	0.266	0.260	0.283	0.231	0.269	0.238	0.272	0.240	0.300	**0.226**	**0.264**	0.259	0.287
Traffic	96	**0.343**	**0.248**	0.373	0.280	0.438	0.291	0.395	0.283	0.351	0.257	0.410	0.282	0.360	0.249	0.593	0.321
	192	**0.362**	0.257	0.390	0.288	0.446	0.293	0.410	0.290	0.364	0.265	0.423	0.287	0.379	**0.256**	0.617	0.336
	336	**0.379**	0.266	0.407	0.299	0.461	0.300	0.414	0.295	0.382	0.273	0.436	0.296	0.392	**0.264**	0.629	0.336
	720	**0.413**	**0.284**	0.438	0.310	0.494	0.318	0.445	0.311	0.420	0.292	0.466	0.315	0.432	0.286	0.640	0.350
	Avg	**0.374**	**0.264**	0.402	0.294	0.460	0.301	0.416	0.295	0.379	0.272	0.434	0.295	0.391	**0.264**	0.620	0.336
Solar-Energy	96	0.171	**0.221**	0.224	0.289	0.223	0.274	0.196	0.261	0.187	0.255	0.191	0.256	**0.168**	0.237	0.178	0.256
	192	0.190	**0.236**	0.244	0.289	0.251	0.290	0.224	0.292	0.200	0.270	0.211	0.273	**0.187**	0.263	0.200	0.268
	336	0.203	**0.248**	0.225	0.291	0.270	0.301	0.240	0.308	0.209	0.276	0.228	0.287	**0.196**	0.260	0.212	0.274
	720	0.222	**0.262**	0.243	0.301	0.271	0.298	0.256	0.321	0.213	0.276	0.236	0.295	**0.205**	0.269	0.211	0.273
	Avg	0.197	**0.242**	0.234	0.293	0.254	0.291	0.229	0.296	0.202	0.269	0.217	0.278	**0.189**	0.257	0.200	0.268

Table 15: Forecasting results on additional benchmark datasets [24] (672-pred-96).

Models	**AutoTimes**		PatchTST		iTransformer		DLinear	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Australian Electricity	**0.150**	**0.228**	0.163	0.242	0.153	0.233	0.167	0.250
Bdg-2 Panther	**0.537**	**0.458**	0.565	0.476	0.546	0.462	0.581	0.499
Oikolab Weather	**0.603**	**0.577**	0.635	0.603	0.630	0.591	0.663	0.611

D.2 Zero-Shot Forecasting

Following the zero-shot forecasting of FPT [15], each experiment comprises the source and target datasets. We train a model on the source dataset and apply the model on the target dataset for predictions directly.

Notably, the zero-shot scenarios are conducted respectively on the subsets (e.g. M4 Monthly → M3 Monthly) and the subsets are divided by the sampling frequency but follow different distributions [25].

For M4 → M3, which means training on M4 and testing on M3, we directly utilize the same model in the short-term forecasting experiments reported in Table 11. Considering different horizons in subsets, for M3 Yearly, M3 Quarterly, and M3 Monthly, we directly employ models trained on corresponding subsets of M4 for testing. As for M3 Others, we test using the model trained on M4 Quarterly to keep the same horizon.

For M3 → M4, similarly, for M4 Yearly, M4 Quarterly, and M4 Monthly, we directly employ models trained on corresponding subsets of M3 for testing. For the remaining subsets, M4 Weekly, M4 Daily, and M4 Hourly, we perform inference using the model trained on M3 Monthly. Table 16 shows the detailed result.

Table 16: Results of zero-shot forecasting. We adopt the same protocol as FPT [49]. M4 → M3 means training forecasters on M4 datasets and evaluating the performance on M3, and vice versa. Results of compared baselines are reported from FPT [49]. Lower SMAPE indicates better performance.

	Method	**AutoTimes**	FPT	DLinear	PatchTST	TimesNet	NSformer	FEDformer	Informer	Reformer
M4 → M3	Yearly	**15.71**	16.42	17.43	15.99	18.75	17.05	16.00	19.70	16.03
	Quarterly	**9.35**	10.13	9.74	9.62	12.26	12.56	9.48	13.00	9.76
	Monthly	14.06	14.10	15.65	14.71	**14.01**	16.82	15.12	15.91	14.80
	Others	5.79	**4.81**	6.81	9.44	6.88	8.13	8.94	13.03	7.53
	Average	**12.75**	13.06	14.03	13.39	14.17	15.29	13.53	15.82	13.37
M3 → M4	Yearly	**13.728**	13.740	14.193	13.966	15.655	14.988	13.887	18.542	15.652
	Quarterly	**10.742**	10.787	18.856	10.929	11.877	11.686	11.513	16.907	11.051
	Monthly	**14.558**	14.630	14.765	14.664	16.165	16.098	18.154	23.454	15.604
	Others	**6.259**	7.081	9.194	7.087	6.863	6.977	7.529	7.348	7.001
	Average	**13.036**	13.125	15.337	13.228	14.553	14.327	15.047	19.047	14.092

D.3 Method Generality

Mainstream LLMs predominantly adopt the decoder-only architecture, and AutoTimes can utilize any decoder-only LLM. We conduct experiments on various types and sizes of LLMs, including GPT-2 [31], multiple sizes of OPT [46], and LLaMA [36]. Detailed configurations and results are shown in Table 17 and 18, demonstrating a general trend where performance improves as the model size increases, consistent with the scaling law [16].

Table 17: Detailed method configurations of AutoTimes for alternative language models.

Base LLM	GPT-2 (124M)	OPT-350M	OPT-1.3B	OPT-2.7B	OPT-6.7B	LLaMA-7B
Hidden Dim.	768	1024	2048	2560	4096	4096
Embedding	2-layer MLP	2-layer MLP	2-layer MLP	2-layer MLP	2-layer MLP	Linear
Trainable Param. (M)	0.44	0.58	1.10	1.36	2.15	0.79

D.4 Variable Lookback Length

In the conventional forecasting paradigm, deep forecasters are trained respectively on lookback/forecast lengths, limiting their applicability to a single lookback length. In contrast, LLMs have the versatility to handle various input lengths. This capability is derived from Rotary Position Embedding [33] and the next toke prediction objective, where LLMs are trained with token-wise supervision in Equation 8, that is, the generated token at each position is supervised. While non-autoregressive LLM4TS methods are typically constrained to a fixed lookback setting, AutoTimes with a consistent training objective has the flexibility to deal with different lookback lengths. We present the results in Figure 8, where we adapt the LLM by AutoTimes with the context length of $C = 672$, and evaluate the performance with different lookback lengths, which demonstrates the inherited versatility of LLM-based forecasters. Moreover, the performance is generally improving with increased available lookback observations, leading to an averaged 9.3% promotion from 384 to 672. By contrast, several works have observed that the performance of respectively trained deep forecasters does not necessarily improve with the increasing of lookback length [22, 26, 45].

Table 18: Full Results of alternative LLMs, which are adapted with the context length $C = 672$.

LLM		GPT-2 (124M)		OPT-350M		OPT-1.3B		OPT-2.7B		OPT-6.7B		LLaMA-7B	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	96	0.140	0.236	0.136	0.233	0.132	0.228	0.132	0.227	0.130	0.226	0.129	0.225
	192	0.159	0.253	0.154	0.249	0.150	0.245	0.149	0.244	0.148	0.242	0.147	0.241
	336	0.177	0.270	0.171	0.267	0.167	0.262	0.167	0.262	0.165	0.260	0.162	0.258
	720	0.216	0.303	0.211	0.301	0.206	0.296	0.207	0.297	0.204	0.295	0.199	0.288
	Avg	0.173	0.266	0.168	0.263	0.164	0.258	0.164	0.258	0.162	0.256	0.159	0.253
ETTh1	96	0.360	0.397	0.365	0.403	0.357	0.395	0.360	0.398	0.357	0.397	0.360	0.400
	192	0.391	0.419	0.395	0.423	0.389	0.417	0.389	0.419	0.386	0.417	0.388	0.419
	336	0.408	0.432	0.411	0.434	0.408	0.431	0.404	0.430	0.404	0.429	0.401	0.429
	720	0.429	0.452	0.432	0.457	0.430	0.452	0.421	0.447	0.427	0.454	0.406	0.440
	Avg	0.397	0.425	0.401	0.429	0.396	0.424	0.394	0.424	0.394	0.424	0.389	0.423
Traffic	96	0.369	0.257	0.371	0.260	0.361	0.253	0.358	0.251	0.357	0.251	0.343	0.248
	192	0.394	0.268	0.393	0.270	0.383	0.263	0.380	0.261	0.379	0.261	0.362	0.257
	336	0.413	0.278	0.411	0.279	0.402	0.273	0.399	0.271	0.398	0.272	0.379	0.266
	720	0.449	0.299	0.446	0.300	0.440	0.295	0.437	0.293	0.437	0.294	0.413	0.284
	Avg	0.406	0.276	0.405	0.277	0.397	0.271	0.394	0.269	0.393	0.270	0.374	0.264
Weather	96	0.158	0.208	0.157	0.208	0.157	0.207	0.157	0.207	0.159	0.209	0.153	0.203
	192	0.207	0.254	0.205	0.252	0.207	0.253	0.206	0.253	0.208	0.256	0.201	0.250
	336	0.262	0.298	0.261	0.294	0.263	0.296	0.265	0.297	0.268	0.302	0.256	0.293
	720	0.342	0.353	0.335	0.346	0.334	0.347	0.344	0.351	0.354	0.360	0.235	0.273
	Avg	0.242	0.278	0.240	0.275	0.240	0.276	0.243	0.277	0.247	0.282	0.235	0.273

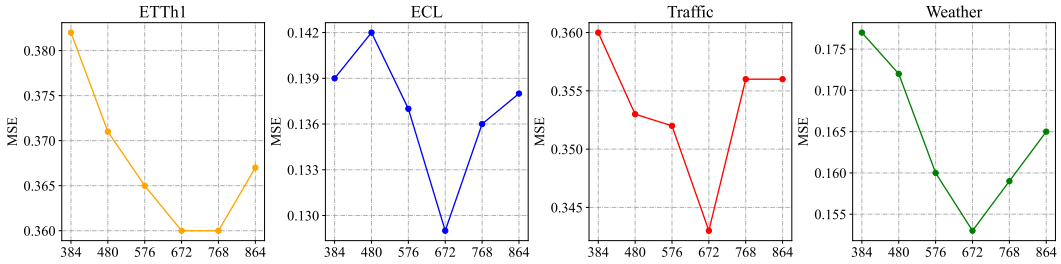

Figure 8: Performance of LLM-based forecasters on the pred-96 scenario, which are adapted by AutoTimes by the context length $C = 672$ and directly applied on different lookback lengths.

D.5 Timestamps as Position Embeddings

We conduct an ablation study on the proposed position embeddings that integrate timestamps, a prevalent textual covariate in real-world applications. As shown in Figure 9, the forecasting performance is consistently promoted across all multivariate datasets and prediction lengths. The steady improvement can be attributed to the fact that timestamps demote the absolute position of time series segments on the timeline, explicitly aligning different variates in multivariate scenarios. The increasing promotion with longer prediction length also implies that chronological information, such as date and periodicity, yields significant benefits for long-term forecasting.

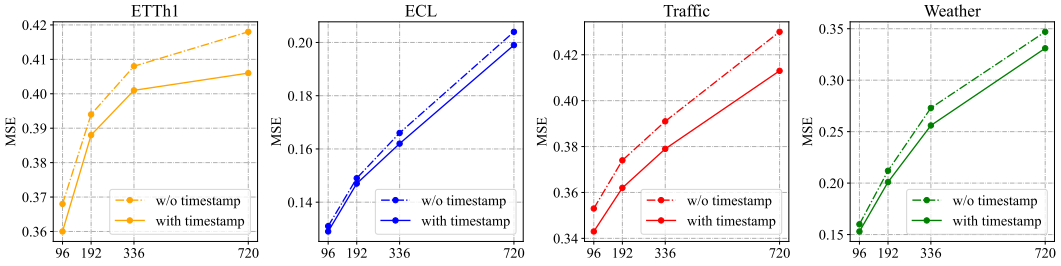

Figure 9: Ablation on whether to utilize textual timestamps as the position embedding. Results of different prediction lengths are provided, where the embedding leads to consistent performance promotion across all datasets, and the promotion can increase with a longer prediction length.

D.6 In-Context Forecasting

For in-context forecasting, similar to zero-shot forecasting in Appendix D.2, we train our model using the source dataset and directly evaluate it on the target dataset. In this task, we first choose M4 as the source dataset and M3 as the target dataset. It is important to note that the structure of the M3 and M4 datasets differs from typical datasets used for long-term forecasting. They consist of multiple univariate time sequences of different lengths. The final part of each sequence serves as the test set, while the preceding part is used for training.

Implementation In zero-shot scenarios, we use a sequence of length F preceding and consecutive to the test set as input, referred to as the lookback window, where F is the forecast length of each subset. During in-context forecasting, we concatenate the first $2F$ time points that belong to the same sequence with the lookback window as input. We aim to enhance prediction performance by incorporating more contextual information. Too short sequences ($\leq 4F$) are discarded to prevent overlap between the prompt and the lookback window. For a fair comparison, both zero-shot and in-context forecasting performance are reported on the same remaining sequences. Figure 12 provides showcases of zero-shot and in-context forecasting.

Prompt engineering Regarding in-context learning, we further delve into the effect of different strategies to retrieve time series as prompts, which is provided in Table 19. **P.1** and **P.2** correspond to the zero-shot and in-context forecasting evaluated in Section 4.3. The prompt of **P.3** contains the last $2F$ time points preceding the beginning of the lookback window. Another retrieval of prompts as **P.4**, adopts time series that come from another uncorrelated time series (out-of-series). We can obtain the following observations:

- **P.1** *v.s.* **P.4** indicates that the selected prompt is not suitable, since the prompt does not come from the earlier observations of the same series to be predicted. Although the context window becomes larger, the averaged performance will deteriorate because of irrelevant prompts.
- **P.2** *and* **P.3** indicates that in most cases, selecting the relevant $2F$ time series from the same series can provide better contextual information.

This highlights the prompt engineering for in-context forecasting. An intuitive suggestion is to utilize consecutive, inter-periodic, and multiple prompts. To verify this idea, we analyze the periodic effect of time series prompts.

Table 19: Effects of different strategies to retrieve time series as prompts for in-context forecasting.

Context for prediction	Yearly	Quarterly	Monthly	Others	Averaged Err.
P.1: Lookback F	21.52	12.03	13.09	8.46	13.61
P.2: Prompt from first $2F$ + Lookback F	17.03	10.29	12.24	5.33	11.80 $\downarrow$
P.3: Prompt from last $2F$ + Lookback F	16.30	9.59	12.09	6.24	11.48 $\downarrow$
P.4: Prompt from $2F$ of other series + Lookback F	18.95	12.18	14.37	9.46	13.98 $\uparrow$

In previous experiments, we adopt M3 and M4 datasets, which are consistent with the zero-shot experiment of FPT [49], to present the promotion of our in-context paradigm. To provide more rigorous conclusions, we extend the evaluation to widely recognized datasets. Details of the experiment are as follows: By using a trained model checkpoint on a source domain (Traffic), we conduct forecasting without gradient update on target ETT datasets. We evaluate the pred-96 performance on the last variate (OT). For the zero-shot scenario, the input is length-288 lookback series. For in-context forecasting, the input is (length-384 series prompt + length-288 lookback series). Considering the dataset periodicity, the prompt is uniformly selected as the Ahead-24 (one-day-ahead) series of the original lookback series. To eliminate the performance boost that comes from extending the input length, we also provide the results of length-672 lookback series in the zero-shot scenario. Moreover, we further delve into the effect of different strategies to select time series prompts:

- **Ahead-Period**: The prompt is uniformly selected as the Ahead-24 series of the original lookback series where 24 is one of the periods (daily period) of ETT.
- **Ahead-Random**: The prompt is randomly selected as the previous series of the original series.
- **Fixed Prompt**: The prompt is fixed as the first 384 time points in the variate-OT.
- **Other Variate**: The prompt is uniformly selected as Ahead-24 series, but comes from other variates.

Results in Table 20 demonstrate the effectiveness of using suitable time series prompts and highlight the influence of prompt engineering. Using inter-period prompts can outperform simply extending lookback window.

The benefit of the proposed in-context forecasting is to extend the input context of time series forecasting beyond a continuous lookback window. Since the essence of prompts is to incorporate useful domain-specific knowledge, here is one use case of in-context forecasting: Considering predicting the weather of one day, one approach is to extend the lookback length from days to weekends. However, it can also introduce noisy information since

Table 20: Strategies to select time series prompts based on periodicity for in-context forecasting.

Context for prediction	ETTh1-OT	ETTh2-OT	ETTm1-OT	ETTm2-OT	Average Err.
P.0: Zero-Shot (Input-288))	0.0673	0.1637	0.0424	0.1669	0.1101
P.1: Zero-Shot (Input-672)	0.0657	0.1538	0.0415	0.1701	0.1078
P.2: Ahead-Period (Input-672)	**0.0645**	**0.1513**	**0.0399**	**0.1629**	**0.1047**
P.3: Ahead-Random (Input-672)	0.0666	0.1621	0.0407	0.1719	0.1103
P.4: Fixed Prompt (Input-672)	0.0769	0.1859	0.0512	0.2104	0.1311
P.5: Other-Variates (Input-672)	0.1263	0.1780	0.0852	0.2297	0.1548

non-stationary meteorological conditions can change with seasons. Another practical way is to consider how the weather changes on the same day in the last year (or years). Although the input is not continuous, the input context becomes more relevant based on prior knowledge about the periodicity (yearly). Therefore, in-context forecasting makes prior knowledge incorporatable and gets performance promotion.

D.7 Ablation Study

In addition to the ablation study of whether LLMs are useful in AutoTimes (Table 6), we further delve into the main difference between our method and previous LLM4TS approach and provide a comprehensive ablation study. The results presented in Table 21 demonstrate that the performance of non-autoregression projection is consistently inferior to that of our autoregressive AutoTimes approach.

Table 21: Ablation study of the autoregression. *FlattenHead* replaces the segment-wise projection of AutoTimes by flatten and linear head [26], which is prevalent in non-autoregressive forecasters.

Dataset	ETTh1				ECL				Weather				Traffic			
Type	**AutoTimes**		FlattenHead		**AutoTimes**		FlattenHead		**AutoTimes**		FlattenHead		**AutoTimes**		FlattenHead	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Pred-96	**0.360**	**0.400**	0.385	0.420	**0.129**	**0.225**	0.142	0.247	**0.153**	**0.203**	0.155	0.209	**0.343**	**0.248**	0.367	0.261
Pred-192	**0.388**	**0.419**	0.445	0.463	**0.147**	**0.241**	0.157	0.259	**0.201**	**0.250**	0.202	0.251	**0.362**	**0.257**	0.391	0.282
Pred-336	**0.401**	**0.429**	0.463	0.475	**0.162**	**0.258**	0.201	0.311	**0.256**	**0.293**	0.257	0.286	**0.379**	**0.266**	0.404	0.287
Pred-720	**0.406**	**0.440**	0.574	0.542	**0.199**	**0.288**	0.232	0.331	**0.331**	**0.345**	0.333	0.261	**0.413**	**0.284**	0.432	0.294

E Showcases

To facilitate a clear comparison among various models, we present additional prediction showcases for long-term forecasting and short-term forecasting. These examples are provided by the following models: TimeLLM [15], FPT [49] and PatchTST [26]. Of all the models, AutoTimes delivers the most accurate future series predictions. Additionally, we provide the showcases of zero-shot and in-context forecasting in Figure 12.

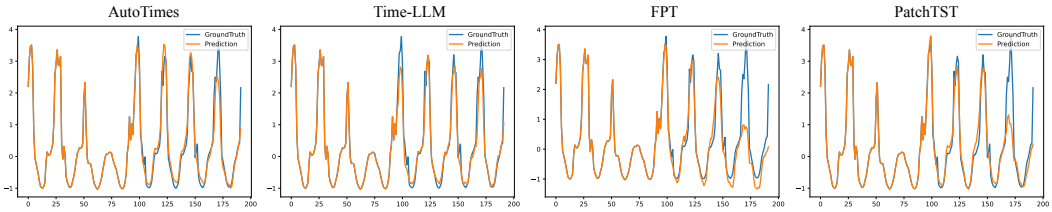

Figure 10: Visualization of input-672-predict-96 results on the Traffic dataset.

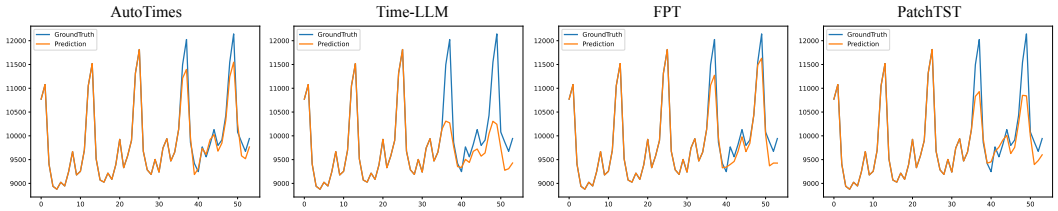

Figure 11: Visualization of input-36-predict-18 results on the M4 Monthly dataset.

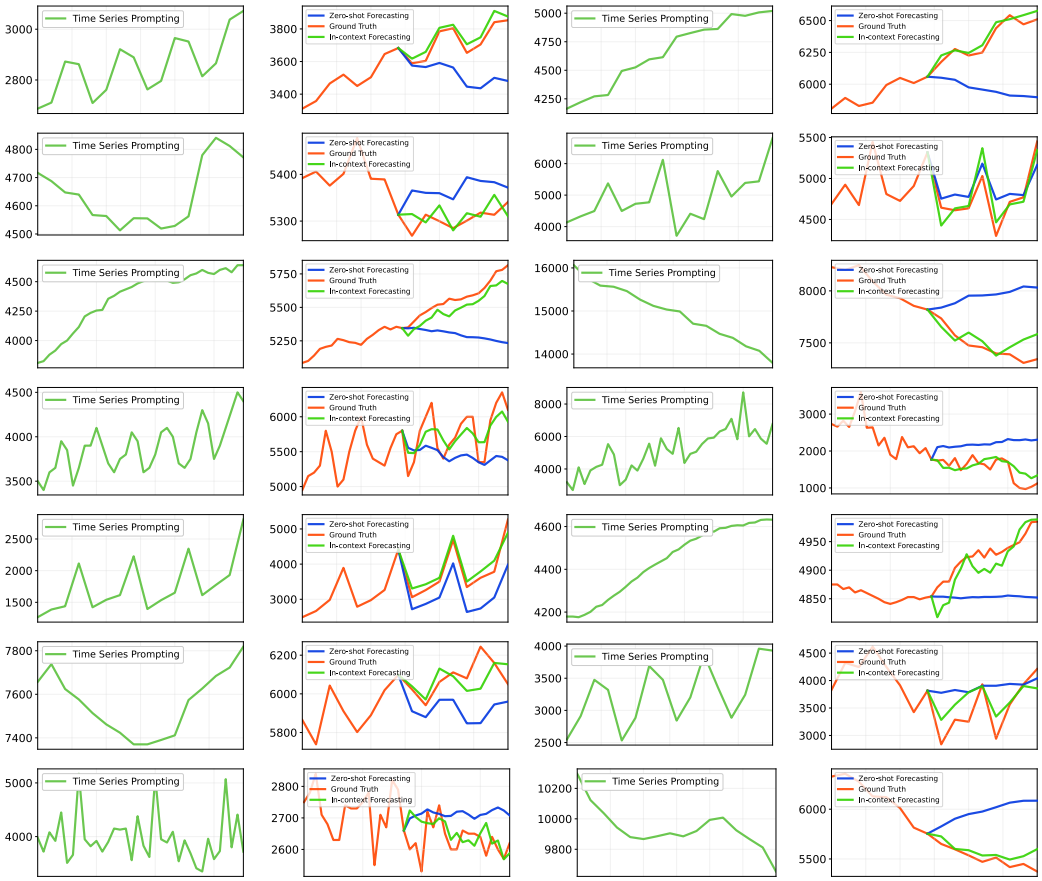

Figure 12: Showcases of zero-shot and in-context forecasting. For in-context forecasting, beyond the lookback window, we uniformly adopt the first $2F$ time points that belong to the same sequence as the prompt and concatenate them as the prediction context, which achieves a more accurate prediction.

F Broader Impact

F.1 Impact on Real-world Applications

This paper copes with general-purpose time series forecasting, which is faced with increasing challenges such as the versatility to handle variable-length scenarios, good generalizability with scarce samples, utilization of multimodality, and instructive downstream prompts. Since previous studies have demonstrated the feasibility of leveraging large language models for time series, we propose a simple but effective approach as AutoTimes to obtain LLM-based forecasters, which keeps the consistency of autoregression. Our model achieves state-of-the-art performance on forecasting benchmarks and demonstrates remarkable adaptation speed and parameter efficiency. Besides, advanced capabilities such as multi-step generation and in-context learning are inherited by the repurposed forecaster. Therefore, the proposed method makes it promising to tackle real-world applications,

which helps our society prevent risks in advance and make better decisions with limited computational budgets. Our paper mainly focuses on scientific research and has no obvious negative social impact.

F.2 Impact on Future Research

In this paper, we find prevalent non-autoregressive LLM4TS methods have inconsistencies in the model structure and generative approach with LLMs, leading to insufficient utilization of the inherent multi-step token transition. Given that the generalizability and generative ability of LLMs are largely derived from the autoregressive manner, the potentials of LLMs may not be fully exhibited in time series forecasting. Therefore, we propose to adapt LLMs by the consistent training objective, the next token prediction, and accomplish arbitrary-length forecasting by iterative generation. Beyond the conventional forecasting paradigm, we propose in-context forecasting, where the context for prediction is extended, and earlier historical time series can be utilized as advantageous prompts. The compatibility with LLMs and insights from autoregression can be instructive for future LLM4TS research and the development of foundation time series models.

G Limitation

The proposed method has not supported probabilistic forecasting, since AutoTimes only establishes the mapping between time series segments to latent embeddings of the LLM, instead of discrete language tokens. Advanced low-rank adaptation is under exploration in our work, which can further align suitable token transitions as the future extrapolation of time series. More deftly designed embedding and projection layers are underexplored to support more compatible tokenization for time series. Besides, it is fascinating to apply AutoTimes on real-world multimodal time series datasets (such as news-stocks, and logs-measurements), which leaves our future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please refer to Section 1 of the main text, where the claims and contributions are included.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to Section G of Appendix, where we provide several aspects of limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Please refer to Section B in the main text and code in our public repository, including the detailed configurations of experiments and the scripts for reproduction.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Please refer to Section A of Appendix and code in our public repository, including dataset descriptions and how to access them.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please refer to Section C and D of Appendix, where we state how hyperparameters are chosen and the detailed description of experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Please refer to Section B and Table 9 of Appendix, where we report standard deviations of results with three random seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Section B of Appendix, where we provide sufficient information on the computing resource.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and the reasearch conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please refer to Section F of Appendix, where we discuss societal impacts and influences on future research.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All creators of datasets are properly credited by citations in Section A.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.