Align and Fine-Tune: Enhancing LLMs for Time-Series Forecasting

Ching Chang[1] **Wei-Yao Wang**[1] **Wen-Chih Peng**[1] **Tien-Fu Chen**[1] **Sagar Samtani**[2]
[1]National Yang Ming Chiao Tung University [2]Indiana University
`blacksnail789521.cs10@nycu.edu.tw`, `sf1638.cs05@nctu.edu.tw`
`{wcpeng, tfchen}@cs.nycu.edu.tw`, `ssamtani@iu.edu`

Abstract

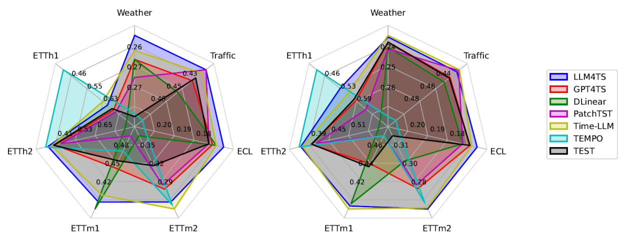
Multivariate time-series forecasting is vital in fields like economic planning and weather prediction, but deep models often require large datasets, limiting their practicality. Pre-trained Large Language Models (LLMs) have been adapted for time-series tasks, but challenges persist due to differences between time-series and linguistic data, and the need for multi-scale temporal processing. To address these challenges, we introduce LLM4TS, a framework that leverages LLMs for time-series forecasting through a two-stage fine-tuning process: *time-series alignment* to adapt LLMs to time-series data and *forecasting fine-tuning* for specific tasks. A novel two-level aggregation method integrates multi-scale temporal data within LLMs. Experiments show that LLM4TS outperforms state-of-the-art methods, excelling in both full-shot and few-shot scenarios. Comparisons with other unsupervised approaches highlight LLM4TS's superior representation learning.

1 Introduction

Forecasting in multivariate time-series analysis is crucial, particularly in applications like economic planning [1, 2, 3] and weather prediction [4, 5, 6]. Deep train-from-scratch models have shown promise in time-series forecasting [7, 8, 9, 10, 4, 11], but they often require large datasets, which are not always available in real-world scenarios [12, 13, 14]. Recent research has turned to pre-trained Large Language Models (LLMs) from NLP [15, 16, 17], leveraging their strong representation learning and few-shot capabilities. These models can be fine-tuned for non-linguistic datasets, including time-series data [18, 19], with minimal data and parameters. However, applying LLMs to time-series forecasting presents challenges due to the unique characteristics of time-series data.

The first challenge is their limited adaptation to the unique characteristics of time-series data, as LLMs are primarily pre-trained on linguistic data. Although LLMs have shown effectiveness in transfer learning across various modalities due to their data-independent self-attention mechanism [18], they struggle to recognize key time-series patterns, leading to inaccuracies in fields like meteorology and energy management [4]. The second challenge lies in LLMs' limited capacity to process multi-scale temporal information, which is crucial for accurate time-series analysis. LLMs typically do not account for different time units and significant events, making it difficult to identify and predict patterns across various time scales [9, 20], resulting in inaccurate forecasting [15, 17].

To address the challenges of adapting LLMs for time-series forecasting, we propose LLM4TS. LLM4TS uses a two-stage fine-tuning process: *time-series alignment* adapts LLMs to time-series data with an autoregressive objective, while *forecasting fine-tuning* targets specific forecasting tasks. Most LLM parameters are frozen, preserving representation learning and ensuring strong performance in both full- and few-shot scenarios. LLM4TS also employs a two-level aggregation strategy, embedding multi-scale temporal information into patches to capture both series values and time-specific context, making it a data-efficient and robust time-series forecaster. The key contributions are:



(a) 5% training data

(b) 10% training data

Figure 1: **Model performance comparison on few-shot forecasting.**

- **Alignment with Time-Series Data**: LLM4TS aligns pre-trained LLMs with time-series characteristics, leveraging their representation learning and few-shot capabilities.
- **Incorporation of Multi-Scale Temporal Data**: A novel two-level aggregation method integrates multi-scale temporal information within LLMs.
- **Superior Forecasting Performance**: LLM4TS outperforms state-of-the-art methods across 7 benchmarks, excelling in few-shot scenarios with minimal data.

2 Method

A sliding window is used to extract sequential samples from a complete, evenly-sampled multivariate time-series. This window, with a stride of 1, spans $T_{in} + T_{out}$, capturing past data $\mathbf{x}_{in} = (d_1, \ldots, d_{T_{in}})$ and future data $\mathbf{x}_{out} = (d_{T_{in}+1}, \ldots, d_{T_{in}+T_{out}})$. Each time step t contains a C-dimensional vector, where C represents the number of features. The objective is to use past data $\mathbf{x}_{in} \in \mathbb{R}^{T_{in} \times C}$ to predict future data $\mathbf{x}_{out} \in \mathbb{R}^{T_{out} \times C}$.

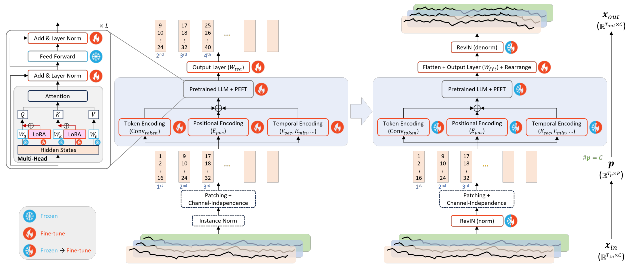
2.1 Time-Series Alignment

Pre-trained LLMs, like GPT-2 [15], are primarily trained on language data, limiting their effectiveness with time-series data. To address this, we introduce a *time-series alignment* stage that adapts LLMs to time-series characteristics using the same autoregressive training method from their original pre-training. As shown in Fig. 2(a), the model takes an input sequence of patched time-series data (e.g., 1st patch, 2nd patch) and generates an output sequence shifted by one patch (e.g., 2nd patch, 3rd patch).

Instance Normalization To ensure stability across modalities, instance normalization is applied alongside layer normalization without a trainable affine transformation, preserving data suitability for the autoregressive objective. Given an input time-series $\mathbf{x}_{in} \in \mathbb{R}^{T_{in} \times C}$, instance normalization produces $\mathbf{x}_{normed} \in \mathbb{R}^{T_{in} \times C}$ with zero mean and unit standard deviation: $\mathbf{x}_{normed} = \text{IN}(\mathbf{x}_{in})$.

Time-Series Tokenization Standard LLM context windows, like the 1024 tokens in GPT-2, are insufficient for long-term time-series forecasting. To address this, we use *channel-independence* and *patching* [8] for tokenization. Channel-independence converts multivariate data into univariate series, reducing the dimension to $\mathbb{R}^{T_{in} \times 1}$. Patching then groups adjacent time steps, reducing the time dimension to T_p and expanding the feature dimension to the patch length P. For a normalized time-series $\mathbf{x}_{normed} \in \mathbb{R}^{T_{in} \times C}$, the process is: $\mathbf{p} = \text{patching}(\text{CI}(\mathbf{x}_{normed}))$.

Three Encodings for Patched Time-Series Data To adapt pre-trained LLMs for time-series data, we introduce three encoding layers: token, positional, and multi-scale temporal. The original token encoding is replaced with a 1D convolutional layer (Conv_{token}) to better handle vectorized time-series data, producing the token embedding $\mathbf{e}_{token} = \text{Conv}_{token}(\mathbf{p})$. Positional encoding uses a trainable lookup table E_{pos} to generate the positional embedding $\mathbf{e}_{pos} = E_{pos}(\mathbf{i})$. To address multi-scale temporal information, a two-level aggregation strategy is used. Level 1 aggregates temporal attributes (seconds, minutes, hours, etc.) within each timestamp using a trainable lookup table E_a. Level 2 applies pooling to select the first timestamp as representative of each patch, creating the temporal



(a) Time-Series Alignment (b) Forecasting Fine-Tuning

Figure 2: **LLM4TS framework.** The numbers in the patched time-series (e.g., 1, 2, ..., 16 in the first patch) represent the sequential timestamps. The framework has two stages: (a) Time-series alignment using an autoregressive approach, and (b) Forecasting fine-tuning, starting with linear probing (only the output layer is unfrozen) and then full fine-tuning (unfreezing all layers and PEFT components).

embedding $\mathbf{e}_{temp} = \text{Pooling}\left(\sum_{a\in\{\text{sec,min,hour,...}\}} E_a(\mathbf{t}_a)\right)$. The final embedding $\mathbf{e}$ is obtained by summing the token, positional, and temporal embeddings: $\mathbf{e} = \mathbf{e}_{token} + \mathbf{e}_{pos} + \mathbf{e}_{temp}$.

Pre-Trained LLM To preserve LLMs' data-independent representation learning, most parameters, especially in multi-head attention and feed-forward layers, are kept fixed, as training from scratch often reduces performance. For the remaining 1.5% of trainable parameters, we use two Parameter-Efficient Fine-Tuning (PEFT) methods: Layer Normalization Tuning, which makes the affine transformation in layer normalization trainable, and Low-Rank Adaptation (LoRA), which adds trainable low-rank matrices in self-attention. The embedding $\mathbf{e}$ is passed through pre-trained Transformer blocks (TBs), producing final embeddings $\mathbf{z} = \text{TBs}(\mathbf{e})$. A linear output layer W_{tsa} then transforms $\mathbf{z}$ back into patched time-series data: $\hat{\mathbf{p}}_{shifted} = \mathbf{z}W_{tsa}^{\top}$. This predicted output aligns with the autoregressive objective by shifting the original patches by one. The Mean Squared Error (MSE) loss function is used to ensure accuracy: $\mathcal{L}_{tsa} = \text{MSE}(\mathbf{p}_{shifted}, \hat{\mathbf{p}}_{shifted})$.

2.2 Forecasting Fine-tuning

After aligning the pre-trained LLM with patched time-series data, the trained weights, including those from the encoding layers, are transferred to the *forecasting fine-tuning* stage. Fine-tuning can be done through full fine-tuning (updating all parameters) or linear probing (updating only the final output layer). A sequential approach, linear probing followed by full fine-tuning (LP-FT), often outperforms either method alone by first optimizing the output layer, then adapting the model for specific tasks, enhancing both out-of-distribution (OOD) robustness and in-distribution (ID) accuracy [21].

The forecasting fine-tuning stage retains the structure from the time-series alignment stage, including the encoding layers and pre-trained LLM, with two key differences: instance normalization and the output layer. Reversible Instance Normalization (RevIN) [22] is introduced to improve forecasting accuracy by normalizing and denormalizing data to address distribution shifts common in time-series data. During tokenization, RevIN is applied before channel-independence and patching: $\mathbf{p} = \text{patching}(\text{CI}(\text{RevIN}_{norm}(\mathbf{x}_{in})))$. Denormalization is only used on unpatched data, so standard instance normalization is used in the alignment stage. The output layer transforms the final embedding $\mathbf{z}$ into the predicted future data by flattening $\mathbf{z}$, passing it through a linear layer W_{fft}, rearranging the data, and applying RevIN's denormalization to produce the final prediction $\hat{\mathbf{x}}_{out} \in \mathbb{R}^{T_{out}\times C}$: $\hat{\mathbf{x}}_{out} = \text{RevIN}_{denorm}(\text{Rearrange}((\text{Flatten}(\mathbf{z}))W_{fft}^{\top}))$. The MSE loss function ensures accurate reconstruction of the future data: $\mathcal{L}_{fft} = \text{MSE}(\mathbf{x}_{out}, \hat{\mathbf{x}}_{out})$.

3 Experiments

3.1 Few-Shot Learning in Long-Term Time-Series Forecasting (LTS)

Tables 1 and 2 show long-term time-series forecasting results using 5% and 10% of the training data, respectively. Both LLM4TS and GPT4TS [18] outperform most train-from-scratch models in few-shot scenarios due to the representation learning capabilities of GPT-2. LLM4TS, with its

Table 1: **Few-shot LTF with 5% training data.**

Methods	LLM4TS		GPT4TS		DLinear		PatchTST		Time-LLM		TEMPO*		TEST	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	**0.256**	**0.292**	0.264	0.302	0.264	0.309	0.270	0.304	0.261	0.309	0.282	0.316	0.283	0.318
ETTh1	0.651	0.551	0.682	0.560	0.750	0.611	0.695	0.569	0.627	0.543	**0.422**	**0.419**	0.674	0.539
ETTh2	0.359	0.405	0.401	0.434	0.828	0.616	0.439	0.448	0.382	0.418	**0.345**	**0.383**	0.393	0.457
ETTm1	0.413	**0.417**	0.472	0.450	**0.401**	**0.417**	0.527	0.476	0.425	0.434	0.501	0.458	0.484	0.449
ETTm2	0.286	0.332	0.308	0.346	0.399	0.426	0.315	0.353	**0.274**	**0.323**	0.281	0.328	0.348	0.343
ECL	**0.173**	**0.266**	0.179	0.273	0.177	0.276	0.181	0.277	0.177	0.268	0.216	0.308	0.181	0.269
Traffic	**0.418**	**0.295**	0.434	0.305	0.451	0.317	0.418	0.297	0.423	0.299	0.492	0.351	0.430	0.320
Avg. Rank	**2.120**	**2.200**	4.200	4.000	4.680	5.080	4.760	4.640	2.720	2.920	4.400	4.280	5.000	4.640

Table 2: **Few-shot LTF with 10% training data.**

Methods	LLM4TS		GPT4TS		DLinear		PatchTST		Time-LLM		TEMPO*		TEST	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	0.235	**0.270**	0.238	0.275	0.241	0.283	0.242	0.279	**0.234**	0.273	0.282	0.316	0.238	0.272
ETTh1	0.525	0.493	0.590	0.525	0.691	0.600	0.633	0.542	0.555	0.522	**0.428**	**0.427**	0.582	0.525
ETTh2	0.366	0.407	0.397	0.421	0.605	0.538	0.415	0.431	0.371	**0.394**	**0.361**	0.398	0.400	0.432
ETTm1	0.408	**0.413**	0.464	0.441	0.411	0.429	0.501	0.466	**0.404**	0.427	0.501	0.458	0.460	0.443
ETTm2	**0.276**	0.324	0.293	0.335	0.316	0.368	0.296	0.343	0.277	**0.323**	0.281	0.328	0.337	0.331
ECL	**0.172**	**0.264**	0.176	0.269	0.180	0.280	0.180	0.273	0.175	0.270	0.216	0.308	0.176	0.269
Traffic	0.432	**0.303**	0.440	0.310	0.447	0.313	0.430	0.305	**0.429**	0.306	0.503	0.358	0.441	0.316
Avg. Rank	**2.036**	**1.679**	4.000	3.786	5.143	5.679	5.036	5.071	2.214	2.786	4.786	4.821	4.536	3.857

additional time-series alignment and multi-scale temporal integration, consistently surpasses GPT4TS across datasets. Among LLM-based methods (Time-LLM, TEMPO, TEST), LLM4TS consistently achieves the best results, particularly on larger datasets, even with only 5% or 10% of the data. Note that TEMPO* is evaluated under a zero-shot setting as it is a prompt-based method.

3.2 Full-Shot Learning in Long-Term Time-Series Forecasting (LTF)

Table 3 presents the results of long-term time-series forecasting averaged across prediction lengths $T_{out} \in \{96, 192, 336, 720\}$. LLM4TS excels not only in few-shot learning but also outperforms all train-from-scratch methods even with full dataset access, thanks to its two-stage fine-tuning and multi-scale temporal integration. In contrast, GPT4TS fails to outperform traditional models in full-shot scenarios due to its lack of time-series alignment and multi-scale temporal integration. Among LLM-based methods, Time-LLM leads in general, but LLM4TS outperforms it in few-shot scenarios, showcasing superior data efficiency and robustness in limited-data settings.

Table 3: **Full-shot LTF.**

Methods	LLM4TS		GPT4TS		DLinear		PatchTST		Time-LLM		TEMPO*		TEST	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	**0.223**	0.260	0.237	0.271	0.249	0.300	0.226	0.264	0.226	**0.258**	0.282	0.316	0.229	0.269
ETTh1	**0.404**	**0.418**	0.428	0.426	0.423	0.437	0.413	0.431	0.408	0.424	0.428	0.427	0.414	0.432
ETTh2	0.333	0.383	0.355	0.395	0.431	0.447	**0.330**	**0.379**	0.334	0.383	0.361	0.398	0.331	0.380
ETTm1	0.343	0.378	0.352	0.383	0.357	0.379	0.351	0.381	**0.329**	**0.372**	0.501	0.458	0.353	0.382
ETTm2	**0.251**	0.313	0.267	0.326	0.267	0.334	0.255	0.315	**0.251**	0.314	0.281	0.328	0.279	**0.291**
ECL	0.159	0.253	0.167	0.263	0.166	0.264	0.162	**0.253**	**0.159**	**0.253**	0.216	0.308	0.163	0.254
Traffic	0.401	0.273	0.414	0.295	0.434	0.295	0.391	0.264	**0.388**	**0.264**	0.503	0.358	0.431	0.295
Avg. Rank	2.036	2.214	4.786	4.750	5.536	5.357	2.571	2.750	**1.643**	**2.143**	6.607	6.250	4.214	3.679

Table 4: **Unsupervised representation learning evaluation in forecasting with linear probing.**

Methods		LLM4TS		PatchTST		BTSF		TS2Vec		TNC		TS-TCC	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	0.315	0.365	0.322	0.369	0.541	0.519	0.599	0.534	0.632	0.596	0.653	0.610
	48	0.342	0.384	0.354	0.385	0.613	0.524	0.629	0.555	0.705	0.688	0.720	0.693
	168	0.401	0.415	0.419	0.424	0.640	0.532	0.755	0.636	1.097	0.993	1.129	1.044
	336	0.421	0.427	0.445	0.446	0.864	0.689	0.907	0.717	1.454	0.919	1.492	1.076
	720	0.426	0.447	0.487	0.478	0.993	0.712	1.048	0.790	1.604	1.118	1.603	1.206
	Avg.	**0.381**	**0.408**	0.405	0.420	0.730	0.595	0.788	0.646	1.098	0.863	1.119	0.926

3.3 Unsupervised Representation Learning

We assess LLM4TS's representation learning using linear evaluation on time-series forecasting, where the model is pre-trained with an autoregressive objective, then frozen, and a linear layer is trained for forecasting. Table 4 shows LLM4TS's superior performance on the ETTh1 dataset, demonstrating its effective adaptation to time-series data during alignment. This evaluation focuses on self-supervised methods, excluding deep train-from-scratch models and GPT4TS, which lacks a distinct representation learning stage. Despite PatchTST's use of an MLM approach for representation learning, LLM4TS outperforms all methods, with a 6.02% average improvement in MSE.

3.4 Ablation Study

We conducted ablation studies to evaluate the impact of key components in LLM4TS, including time-series alignment, multi-scale temporal encoding, and PEFT methods, across both full- and few-shot scenarios. The results highlight the importance of each component in enhancing forecasting accuracy. We also examined various training strategies in the forecasting fine-tuning stage and assessed the benefits of retaining LLMs' pre-trained weights. Detailed findings are in Appendix B.7.

4 Conclusion

This paper present LLM4TS, a framework for time-series forecasting using pre-trained LLMs. LLM4TS employs a two-stage fine-tuning strategy: *time-series alignment* to adapt LLMs to time-series data and *forecasting fine-tuning* for specific tasks. It also introduces a two-level aggregation method to integrate multi-scale temporal data, enhancing LLMs' interpretation of time-related information. Experiments on seven datasets show LLM4TS outperforms state-of-the-art methods, including those trained from scratch, in both full and few-shot scenarios. Future work will explore more recent LLMs like GPT-3.5 and LLaMA-2 to assess advancements, and extend LLM4TS to tasks like classification and anomaly detection to broaden its applicability.

References

[1] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *SIGIR*, pages 95–104. ACM, 2018.

[2] Hiteshi Tandon, Prabhat Ranjan, Tanmoy Chakraborty, and Vandana Suhag. Coronavirus (covid-19): Arima-based time-series analysis to forecast near future and the effect of school reopening in india. *Journal of Health Management*, 24(3):373–388, 2022.

[3] Ananda Chatterjee, Hrisav Bhowmick, and Jaydip Sen. Stock price prediction using time series, econometric, machine learning, and deep learning models. *CoRR*, abs/2111.01137, 2021.

[4] 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 *AAAI*, pages 11106–11115. AAAI Press, 2021.

[5] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[6] Pradeep Hewage, Ardhendu Behera, Marcello Trovati, Ella Pereira, Morteza Ghahremani, Francesco Palmieri, and Yonghuai Liu. Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station. *Soft Comput.*, 24(21):16453–16482, 2020.

[7] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI*, pages 11121–11128. AAAI Press, 2023.

[8] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *ICLR*. OpenReview.net, 2023.

[9] 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.

[10] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 27268–27286. PMLR, 2022.

[11] Kun Yi, Qi Zhang, Wei Fan, Shoujin Wang, Pengyang Wang, Hui He, Ning An, Defu Lian, Longbing Cao, and Zhendong Niu. Frequency-domain mlps are more effective learners in time series forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

[12] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022.

[13] Yian Zhang, Alex Warstadt, Xiaocheng Li, and Samuel R. Bowman. When do you need billions of words of pretraining data? In *ACL/IJCNLP (1)*, pages 1112–1125. Association for Computational Linguistics, 2021.

[14] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022.

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

[16] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[18] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *NeurIPS*, 2023.

[19] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. *CoRR*, abs/2310.01728, 2023.

[20] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *ICLR*. OpenReview.net, 2023.

[21] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *ICLR*. OpenReview.net, 2022.

[22] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *ICLR*. OpenReview.net, 2022.

[23] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Frozen pretrained transformers as universal computation engines. In *AAAI*, pages 7628–7636. AAAI Press, 2022.

[24] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[25] Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Text-to-audio generation using instruction guided latent diffusion model. In *ACM Multimedia*, pages 3590–3598. ACM, 2023.

[26] Fangxun Shu, Lei Zhang, Hao Jiang, and Cihang Xie. Audio-visual LLM for video understanding. *CoRR*, abs/2312.06720, 2023.

[27] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David A. Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, pages 5549–5581. PMLR, 2023.

[28] Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, page 645–654, New York, NY, USA, 2024. Association for Computing Machinery.

[29] Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 11398–11442. PMLR, 2023.

[30] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[31] 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. In *The Twelfth International Conference on Learning Representations*, 2024.

[32] Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. TEST: Text prototype aligned embedding to activate LLM's ability for time series. In *The Twelfth International Conference on Learning Representations*, 2024.

[33] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *ICLR*. OpenReview.net, 2023.

[34] Sangwon Lee, Junho Hong, Ling Liu, and Wonik Choi. Ts-fastformer: Fast transformer for time-series forecasting. *ACM Trans. Intell. Syst. Technol.*, 15(2), feb 2024.

[35] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. In *ICLR*. OpenReview.net, 2022.

[36] Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. Gated transformer networks for multivariate time series classification. *CoRR*, abs/2103.14438, 2021.

[37] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. In *IJCAI*, pages 6778–6786. ijcai.org, 2023.

[38] Sabeen Ahmed, Ian E. Nielsen, Aakash Tripathi, Shamoon Siddiqui, Ravi Prakash Ramachandran, and Ghulam Rasool. Transformers in time-series analysis: A tutorial. *Circuits Syst. Signal Process.*, 42(12):7433–7466, 2023.

[39] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *AAAI*, pages 8980–8987. AAAI Press, 2022.

[40] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. In *ICLR*. OpenReview.net, 2021.

[41] Ling Yang and Shenda Hong. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 25038–25054. PMLR, 2022.

[42] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. In *IJCAI*, pages 2352–2359. ijcai.org, 2021.

Table 5: **Statistical overview of the 7 datasets for long-term time-series forecasting.**

Datasets	Features	Timesteps	Granularity
Weather	21	52,696	10 min
Traffic	862	17,544	1 hour
Electricity	321	26,304	1 hour
ETTh1 & ETTh2	7	17,420	1 hour
ETTm1 & ETTm2	7	69,680	5 min

A Related Work

A.1 Transfer Learning Across Various Modalities with LLMs

Large Language Models (LLMs) have proven highly effective in transfer learning across multiple modalities, including images [23, 24], audio [25, 26], tabular data [27, 28], and time-series data [18, 19]. A key advantage of using LLMs in these different domains is their ability to perform well even with limited data [18]. To maintain the data-independent representation learning capability of LLMs, the majority of their parameters are kept fixed during the transfer learning process. Empirical evidence [23, 18] shows that LLMs with fixed parameters often outperform models trained from scratch, highlighting the benefit of retaining the pre-trained representation learning strengths of these models (additional experiments are discussed in Section B.7.3). Theoretically, it has been demonstrated that the self-attention modules in pre-trained transformers can develop data-independent operations, similar to principal component analysis [18], enabling them to serve as *universal compute engines* [23] or *general computation calculators* [29]. In the time-series domain, GPT4TS [18] utilizes the pre-trained GPT-2, showing strong performance in time-series forecasting, particularly in few-shot conditions, without the need for fine-tuning most parameters. Time-LLM [30] reprograms LLMs for time-series forecasting by transforming time series into textual prototypes and using prompts for guidance, outperforming specialized models in few-shot and zero-shot settings. TEMPO [31] adapts GPT-like models for time-series forecasting by decomposing trends and utilizing prompts for better distribution adaptation, excelling in zero-shot scenarios. TEST [32] aligns time-series data with LLM embeddings through tokenization and contrastive learning, enabling efficient time-series forecasting without fine-tuning. With LLM4TS, we tackle the challenges of adapting LLMs to time-series-specific features and processing multi-scale temporal information, thereby boosting performance in time-series forecasting tasks.

A.2 Long-Term Time-Series Forecasting

Significant efforts have focused on using Transformer models for long-term time-series forecasting [4, 9, 10, 8, 33, 34]. While Transformer-based models have gained prominence, DLinear [7] has shown that a simple linear model can outperform many sophisticated Transformer-based methods. These deep models excel when trained on large datasets, but their performance often diminishes in limited-data settings. LLM4TS, however, achieves new benchmarks alongside state-of-the-art models in both full-data and few-shot conditions.

A.3 Time-Series Representation Learning

In the time-series field, self-supervised learning has emerged as a powerful method for representation learning. Although Transformers are increasingly seen as suitable for end-to-end time-series analysis [35, 36, 8, 37, 38], CNN-based [39] and RNN-based [40] models still dominate in time-series self-supervised learning. However, the ability of Transformers to model long-range dependencies and capture complex patterns makes them well-suited for time-series data, where sequential relationships are intricate. Since the *time-series alignment* phase in LLM4TS can be viewed as a self-supervised learning strategy, we evaluate LLM4TS's capacity for representation learning, demonstrating that Transformers can surpass CNN and RNN-based models in unsupervised time-series learning.

B More on Experiments

B.1 Datasets

In our long-term forecasting experiments, we use seven real-world, publicly accessible benchmark datasets. Table 5 provides detailed statistics for each dataset, including the number of features, the total length, and their sampling frequencies.

Weather[1] consists of local climatological data spanning four years across approximately 1,600 U.S. locations. Each record includes 11 weather variables, alongside the target variable 'wet bulb.' **Traffic**[2] contains hourly data from the California Department of Transportation, capturing road occupancy rates via sensors on freeways in the San Francisco Bay area. **Electricity**[3] comprises hourly power usage data from 321 customers between 2012 and 2014, with 'MT_320' used as the target variable. **ETT** [4] provides long-term electric power deployment data. The dataset includes two hourly-sampled (ETTh1, ETTh2) and two 15-minute-sampled (ETTm1, ETTm2) datasets, spanning over two years from various provinces in China. Each ETT dataset contains one oil temperature variable and six power load variables.

B.2 Evaluation Metrics

In time-series forecasting, the most commonly used evaluation metrics are Mean Squared Error (MSE) and Mean Absolute Error (MAE). The Mean Squared Error (MSE) is defined as:

$$\text{MSE} = \frac{1}{N}\sum_{n=1}^{N}(x_{\text{out}} - \hat{x}_{\text{out}})^2, \tag{1}$$

where x_{out} is the actual future value corresponding to the input past data x_{in}, and $\hat{x}_{\text{out}}$ is the predicted future value based on the input. N represents the total number of samples. The Mean Absolute Error (MAE) is calculated as:

$$\text{MAE} = \frac{1}{N}\sum_{n=1}^{N}|x_{\text{out}} - \hat{x}_{\text{out}}|. \tag{2}$$

B.3 Baselines

For long-term time-series forecasting, we evaluate several state-of-the-art models. The same models are used for few-shot learning and ablation studies. **GPT4TS** [18] uses patching and channel independence to convert time-series data into tokens, leveraging a pre-trained GPT-2 while maintaining most of the pre-trained weights. **DLinear** [7] challenges the dominance of Transformer models in time-series forecasting by introducing a simple single-layer linear model that outperforms complex Transformer-based models on multiple real-world datasets. **PatchTST** [8] employs a Transformer-based approach with patching and channel-independence to convert time-series data into patches for efficient processing. **FEDformer** [10] enhances Transformer models by incorporating seasonal-trend decomposition and frequency analysis for improved efficiency and effectiveness in long-term forecasting, offering linear complexity while capturing global trends and detailed structures. **Time-LLM** [30] reprograms LLMs for time-series forecasting by transforming time-series data into text-based prototypes and using prompts to guide predictions, achieving strong performance in few-shot and zero-shot settings. **TEMPO** [31] adapts GPT-like models for time-series by decomposing trends and using prompts to adjust for distributional changes, excelling in zero-shot environments. **TEST** [32] aligns time-series data with LLM embeddings through tokenization and contrastive learning, allowing effective time-series forecasting without fine-tuning pre-trained LLMs.

For unsupervised representation learning, we consider advanced models that extract meaningful representations from time-series data without relying on labeled data. **PatchTST** [8], in this context, uses a Masked Language Model (MLM) approach, similar to BERT, to learn effective representations for time-series. **BTSF** [41] introduces a Bilinear Temporal-Spectral Fusion framework that integrates

[1]https://www.ncei.noaa.gov/data/local-climatological-data/

[2]http://pems.dot.ca.gov/

[3]https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014

Table 6: **Few-shot long-term forecasting using 5% of the training data.** For most datasets, results are reported over prediction lengths $T_{out} \in \{96, 192, 336, 720\}$. However, for datasets marked with * (ETTh1, ETTh2, and Traffic), only $T_{out} \in \{96, 192, 336\}$ are used because there are insufficient data to constitute a training set when $T_{out} = 720$. The best results are in **bold**, while the second-best results are in underlined. Note that TEMPO* is evaluated under a zero-shot setting as it is a prompt-based method.

Methods		LLM4TS		GPT4TS		DLinear		PatchTST		Time-LLM		TEMPO*		TEST	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.173	0.227	0.175	0.230	0.184	0.242	**0.171**	**0.224**	0.172	0.263	0.211	0.254	0.182	0.276
	192	**0.218**	**0.265**	0.227	0.276	0.228	0.283	0.230	0.277	0.224	0.271	0.254	0.298	0.273	0.283
	336	**0.276**	**0.310**	0.286	0.322	0.279	0.322	0.294	0.326	0.282	0.321	0.292	0.332	0.294	0.325
	720	**0.355**	**0.366**	0.366	0.379	0.364	0.388	0.384	0.387	0.366	0.381	0.370	0.379	0.383	0.388
ETTh1	96	0.509	0.484	0.543	0.506	0.547	0.503	0.557	0.519	0.483	0.464	**0.400**	**0.406**	0.531	0.447
	192	0.717	0.581	0.748	0.580	0.720	0.604	0.711	0.570	0.629	0.540	**0.426**	**0.421**	0.750	0.533
	336	0.728	0.589	0.754	0.595	0.984	0.727	0.816	0.619	0.768	0.626	**0.441**	**0.430**	0.741	0.636
	720	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ETTh2	96	0.314	0.375	0.376	0.421	0.442	0.456	0.401	0.421	0.336	0.397	**0.301**	**0.353**	0.368	0.457
	192	0.365	0.408	0.418	0.441	0.617	0.542	0.452	0.455	0.406	0.425	**0.355**	**0.389**	0.407	0.486
	336	0.398	0.432	0.408	0.439	1.424	0.849	0.464	0.469	0.405	0.432	**0.379**	**0.408**	0.402	0.428
	720	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ETTm1	96	0.349	0.379	0.386	0.405	0.332	**0.374**	0.399	0.414	**0.316**	0.377	0.438	0.424	0.340	0.381
	192	0.374	0.394	0.440	0.438	**0.358**	**0.390**	0.441	0.436	0.450	0.464	0.461	0.432	0.473	0.451
	336	0.411	0.417	0.485	0.459	**0.402**	**0.416**	0.499	0.467	0.450	0.424	0.515	0.467	0.519	0.464
	720	0.516	0.479	0.577	0.499	0.511	0.489	0.767	0.587	**0.483**	**0.471**	0.591	0.509	0.604	0.499
ETTm2	96	0.192	0.273	0.199	0.280	0.236	0.326	0.206	0.288	**0.174**	**0.261**	0.185	0.267	0.254	0.275
	192	0.249	0.309	0.256	0.316	0.306	0.373	0.264	0.324	**0.215**	0.287	0.243	0.304	0.265	**0.286**
	336	0.301	0.342	0.318	0.353	0.380	0.423	0.334	0.367	**0.273**	**0.330**	0.309	0.345	0.360	0.373
	720	0.402	0.405	0.460	0.436	0.674	0.583	0.454	0.432	0.433	0.412	**0.386**	**0.395**	0.511	0.439
ECL	96	**0.139**	**0.235**	0.143	0.241	0.150	0.251	0.145	0.244	0.147	0.242	0.178	0.276	0.144	0.246
	192	**0.155**	0.249	0.159	0.255	0.163	0.263	0.163	0.260	0.158	**0.241**	0.198	0.293	0.180	0.248
	336	**0.174**	**0.269**	0.179	0.274	0.175	0.278	0.183	0.281	0.178	0.277	0.209	0.309	0.194	0.304
	720	0.222	0.310	0.233	0.323	0.219	0.311	0.233	0.323	0.224	0.312	0.279	0.355	**0.205**	**0.277**
Traffic	96	**0.401**	**0.285**	0.419	0.298	0.427	0.304	0.404	0.286	0.414	0.291	0.476	0.343	0.443	0.317
	192	0.418	0.293	0.434	0.305	0.447	0.315	0.412	0.294	0.419	**0.291**	0.496	0.355	**0.407**	0.320
	336	**0.436**	**0.308**	0.449	0.313	0.478	0.333	0.439	0.310	0.437	0.314	0.503	0.356	0.440	0.323
	720	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Avg. Rank		**2.120**	**2.200**	4.200	4.000	4.680	5.080	4.760	4.640	2.720	2.920	4.400	4.280	5.000	4.640

temporal and spectral information, improving time-series representation learning through instance-level augmentation and fusion techniques. **TS2Vec** [39] is a universal framework that learns time-series representations by distinguishing multi-scale contextual information at both the instance and timestamp levels, showing effectiveness across diverse time-series tasks. **TNC** [40] applies the Augmented Dickey-Fuller test to detect temporal neighborhoods and employs Positive-Unlabeled learning to reduce sampling bias. **TS-TCC** [42] generates two distinct views through strong and weak augmentations and improves representations through contrastive learning by focusing on temporal and contextual differences between these views.

B.4 Implementation Details

In our experiments for long-term time-series forecasting, few-shot learning, and ablation studies, we adopt the settings from PatchTST [8] to ensure consistent comparisons across models. We first evaluate the model's performance in few-shot scenarios, followed by a thorough evaluation under full-shot conditions to provide a well-rounded analysis. The look-back window length T_{in} is set to either 336 or 512 (depending on which yields the best results), while the patch length P is set to

Table 7: **Few-shot long-term forecasting using** 10% **of the training data.** We use prediction lengths $T \in \{96, 192, 336, 720\}$ for all datasets. The best results are in **bold**, while the second-best results are underlined. Note that TEMPO* is evaluated under a zero-shot setting as it is a prompt-based method.

Methods		LLM4TS		GPT4TS		DLinear		PatchTST		Time-LLM		TEMPO*		TEST	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.158**	**0.207**	0.163	0.215	0.171	0.224	0.165	0.215	0.161	0.210	0.211	0.254	0.163	0.213
	192	**0.204**	0.249	0.210	0.254	0.215	0.263	0.210	0.257	**0.204**	**0.248**	0.254	0.298	0.230	0.263
	336	**0.254**	0.288	0.256	0.292	0.258	0.299	0.259	0.297	0.261	0.302	0.292	0.332	0.258	**0.282**
	720	0.322	0.336	0.321	0.339	0.320	0.346	0.332	0.346	0.309	0.332	0.370	0.379	**0.301**	**0.328**
ETTh1	96	0.417	0.432	0.458	0.456	0.492	0.495	0.516	0.485	0.448	0.460	**0.400**	**0.406**	0.455	0.457
	192	0.469	0.468	0.570	0.516	0.565	0.538	0.598	0.524	0.484	0.483	**0.426**	**0.421**	0.572	0.519
	336	0.505	0.499	0.608	0.535	0.721	0.622	0.657	0.550	0.589	0.540	**0.441**	**0.430**	0.578	0.531
	720	0.708	0.572	0.725	0.591	0.986	0.743	0.762	0.610	0.700	0.604	**0.443**	**0.451**	0.723	0.594
ETTh2	96	0.282	0.351	0.331	0.374	0.357	0.411	0.353	0.389	**0.275**	**0.326**	0.301	0.353	0.332	0.374
	192	0.364	0.400	0.402	0.411	0.569	0.519	0.403	0.414	0.374	**0.373**	**0.355**	0.389	0.401	0.433
	336	**0.374**	0.416	0.406	0.433	0.671	0.572	0.426	0.441	0.406	0.429	0.379	**0.408**	0.408	0.440
	720	0.445	0.461	0.449	0.464	0.824	0.648	0.477	0.480	0.427	0.449	**0.409**	**0.440**	0.459	0.480
ETTm1	96	0.360	**0.388**	0.390	0.404	0.352	0.392	0.410	0.419	**0.346**	**0.388**	0.438	0.424	0.392	0.401
	192	0.386	**0.401**	0.429	0.423	0.382	0.412	0.437	0.434	**0.373**	0.416	0.461	0.432	0.423	0.426
	336	0.415	**0.417**	0.469	0.439	0.419	0.434	0.476	0.454	**0.413**	0.426	0.515	0.467	0.471	0.444
	720	**0.470**	**0.445**	0.569	0.498	0.490	0.477	0.681	0.556	0.485	0.476	0.591	0.509	0.552	0.501
ETTm2	96	0.184	0.265	0.188	0.269	0.213	0.303	0.191	0.274	**0.177**	**0.261**	0.185	0.267	0.233	0.262
	192	**0.240**	**0.301**	0.251	0.309	0.278	0.345	0.252	0.317	0.241	0.314	0.243	0.304	0.303	0.302
	336	0.294	0.337	0.307	0.346	0.338	0.385	0.306	0.353	**0.274**	**0.327**	0.309	0.345	0.359	0.341
	720	**0.386**	0.393	0.426	0.417	0.436	0.440	0.433	0.427	0.417	**0.390**	**0.386**	0.395	0.452	0.419
ECL	96	**0.135**	**0.231**	0.139	0.237	0.150	0.253	0.140	0.238	0.139	0.241	0.178	0.276	0.138	0.235
	192	0.152	**0.246**	0.156	0.252	0.164	0.264	0.160	0.255	**0.151**	0.248	0.198	0.293	0.158	0.255
	336	0.173	**0.267**	0.175	0.270	0.181	0.282	0.180	0.276	**0.169**	0.270	0.209	0.309	0.176	0.275
	720	0.229	0.312	0.233	0.317	**0.223**	0.321	0.241	0.323	0.240	0.322	0.279	0.355	0.230	**0.311**
Traffic	96	**0.402**	**0.288**	0.414	0.297	0.419	0.298	0.403	0.289	0.418	0.291	0.476	0.343	0.415	0.317
	192	0.416	**0.294**	0.426	0.301	0.434	0.305	0.415	0.296	**0.414**	0.296	0.496	0.355	0.425	0.300
	336	0.429	**0.302**	0.434	0.303	0.449	0.313	0.426	0.304	**0.421**	0.311	0.503	0.356	0.436	0.310
	720	0.480	**0.326**	0.487	0.337	0.484	0.336	0.474	0.331	**0.462**	0.327	0.538	0.376	0.489	0.338
Avg. Rank		**2.036**	**1.679**	4.000	3.786	5.143	5.679	5.036	5.071	2.214	2.786	4.786	4.821	4.536	3.857

16 with a stride S of 8. For unsupervised representation learning, we adjust the settings slightly to $T_{in} = 512$, $P = 12$, and $S = 12$. Following the configuration used in GPT4TS [18], we utilize only the first six layers of the 12-layer GPT-2 base model [15].

B.5 Few-Shot Learning in Long-Term Time-Series Forecasting

Table 6 presents the results of long-term time-series forecasting using only 5% of the training data, while Table 7 shows similar results with 10% of the training data. In both cases, we maintain consistent splits for the training, validation, and test sets across full and few-shot learning scenarios. The training data percentages were deliberately limited to 5% and 10% to assess model performance in few-shot settings. For each dataset, we train a single model during the time-series alignment phase, which is then applied consistently across all prediction lengths. In the forecasting fine-tuning phase, however, we fine-tune a separate model for each prediction length, ensuring that all models share the same hyperparameters.

Both LLM4TS and GPT4TS [18] consistently outperform most models trained from scratch in limited-data scenarios across various datasets, thanks to the pre-trained representation learning capabilities of GPT-2. By incorporating time-series alignment and multi-scale temporal information, LLM4TS proves to be a more data-efficient time-series forecaster than GPT4TS, achieving better

Table 8: **Long-term forecasting for multivariate time-series data.** We use prediction lengths $T \in \{96, 192, 336, 720\}$ for all datasets. The best results are in **bold**, while the second-best results are underlined. Note that TEMPO* is evaluated under a zero-shot setting as it is a prompt-based method.

Methods		LLM4TS		GPT4TS		DLinear		PatchTST		Time-LLM		TEMPO*		TEST	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.147**	**0.196**	0.162	0.212	0.176	0.237	0.149	0.198	**0.147**	0.201	0.211	0.254	0.150	0.202
	192	0.191	0.238	0.204	0.248	0.220	0.282	0.194	0.241	**0.189**	**0.234**	0.254	0.298	0.198	0.246
	336	**0.241**	**0.277**	0.254	0.286	0.265	0.319	0.245	0.282	0.262	0.279	0.292	0.332	0.245	0.286
	720	0.313	0.329	0.326	0.337	0.333	0.362	0.314	0.334	**0.304**	**0.316**	0.370	0.379	0.324	0.342
ETTh1	96	0.371	0.394	0.376	0.397	0.375	0.399	0.370	0.399	**0.362**	**0.392**	0.400	0.406	0.372	0.400
	192	0.403	**0.412**	0.416	0.418	0.405	0.416	0.413	0.421	**0.398**	0.418	0.426	0.421	0.414	0.422
	336	**0.420**	**0.422**	0.442	0.433	0.439	0.443	0.422	0.436	0.430	0.427	0.441	0.430	0.422	0.437
	720	**0.422**	**0.444**	0.477	0.456	0.472	0.490	0.447	0.466	0.442	0.457	0.443	0.451	0.447	0.467
ETTh2	96	0.269	0.332	0.285	0.342	0.289	0.353	0.274	0.336	**0.268**	**0.328**	0.301	0.353	0.275	0.338
	192	**0.328**	0.377	0.354	0.389	0.383	0.418	0.339	0.379	0.329	**0.375**	0.355	0.389	0.340	0.379
	336	0.353	0.396	0.373	0.407	0.448	0.465	**0.329**	**0.380**	0.368	0.409	0.379	0.408	**0.329**	0.381
	720	0.383	0.425	0.406	0.441	0.605	0.551	0.379	0.422	**0.372**	**0.420**	0.409	0.440	0.381	0.423
ETTm1	96	0.285	0.343	0.292	0.346	0.299	0.343	0.290	0.342	**0.272**	**0.334**	0.438	0.424	0.293	0.346
	192	0.324	0.366	0.332	0.372	0.335	0.365	0.332	0.369	**0.310**	**0.358**	0.461	0.432	0.332	0.369
	336	0.353	0.385	0.366	0.394	0.369	0.386	0.366	0.392	**0.352**	**0.384**	0.515	0.467	0.368	0.392
	720	0.408	0.419	0.417	0.421	0.425	0.421	0.416	0.420	**0.383**	**0.411**	0.591	0.509	0.418	0.420
ETTm2	96	0.165	0.254	0.173	0.262	0.167	0.269	0.165	0.255	**0.161**	0.253	0.185	0.267	0.193	**0.237**
	192	0.220	0.292	0.229	0.301	0.224	0.303	0.220	0.292	**0.219**	0.293	0.243	0.304	0.257	**0.264**
	336	**0.268**	0.326	0.286	0.341	0.281	0.342	0.274	0.329	0.271	0.329	0.309	0.345	0.289	**0.295**
	720	**0.350**	0.380	0.378	0.401	0.397	0.421	0.362	0.385	0.352	0.379	0.386	0.395	0.375	**0.369**
ECL	96	**0.128**	0.223	0.139	0.238	0.140	0.237	0.129	**0.222**	0.131	0.224	0.178	0.276	0.132	0.223
	192	**0.146**	**0.240**	0.153	0.251	0.153	0.249	0.157	**0.240**	0.152	0.241	0.198	0.293	0.158	0.241
	336	0.163	0.258	0.169	0.266	0.169	0.267	0.163	0.259	**0.160**	**0.248**	0.209	0.309	0.163	0.260
	720	0.200	0.292	0.206	0.297	0.203	0.301	0.197	**0.290**	**0.192**	0.298	0.279	0.355	0.199	0.291
Traffic	96	0.372	0.259	0.388	0.282	0.410	0.282	**0.360**	0.249	0.362	**0.248**	0.476	0.343	0.407	0.282
	192	0.391	0.265	0.407	0.290	0.423	0.287	0.379	0.256	**0.374**	**0.247**	0.496	0.355	0.423	0.287
	336	0.405	0.275	0.412	0.294	0.436	0.296	0.392	**0.264**	**0.385**	0.271	0.503	0.356	0.430	0.296
	720	0.437	0.292	0.450	0.312	0.466	0.315	0.432	**0.286**	**0.430**	0.288	0.538	0.376	0.463	0.315
Avg. Rank		2.036	2.214	4.786	4.750	5.536	5.357	2.571	2.750	**1.643**	**2.143**	6.607	6.250	4.214	3.679

performance across all datasets. Notably, LLM4TS with just 5% of the training data outperforms the best baseline using 10% of the data. For the largest dataset, Traffic, PatchTST is the top-performing model in the full-shot scenario, but this trend does not hold in few-shot settings. With only 10% of the training data, LLM4TS outperforms PatchTST in 5 out of 8 evaluations, and with just 5% of the data, it leads in 5 out of 6 evaluations. This highlights that, in few-shot scenarios, traditional deep models trained from scratch generally lag behind those leveraging pre-trained LLMs.

We also compare the performance of the latest LLM-based methods for time-series forecasting: Time-LLM, TEMPO, and TEST. TEMPO, a prompt-based method, is evaluated in a zero-shot setting, so its performance remains constant across both few-shot and full-shot scenarios. For larger datasets such as Weather, Electricity, and Traffic, LLM4TS consistently delivers the best results, even with just 5% or 10% of the training data. On the ETTh1 and ETTh2 datasets, TEMPO performs best, while Time-LLM excels on the ETTm1 and ETTm2 datasets. Overall, LLM4TS demonstrates superior performance across these LLM-based methods in both the 5% and 10% training data scenarios.

B.6 Full-Shot Learning in Long-Term Time-Series Forecasting

Table 8 shows the results of long-term time-series forecasting, averaged over a consistent prediction length set $T_{out} \in \{96, 192, 336, 720\}$ for all datasets. Although the primary focus of pre-trained

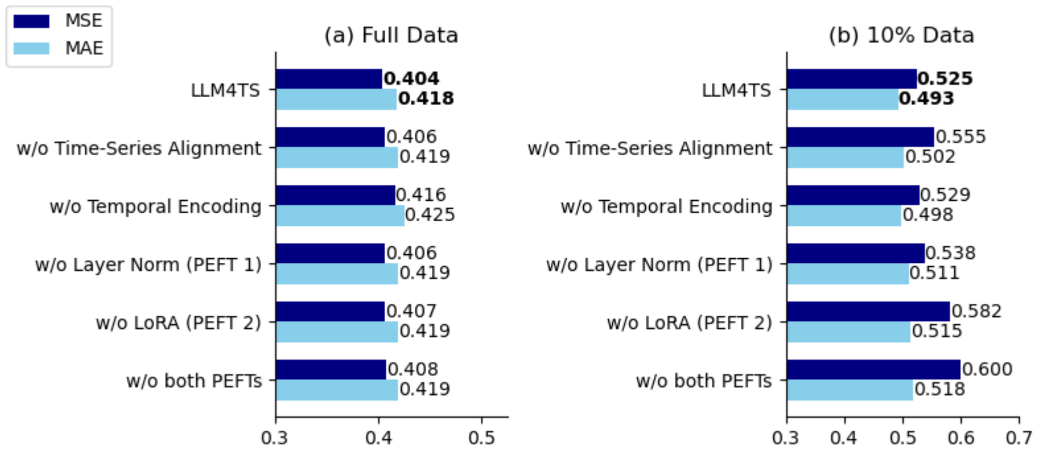


Figure 3: **Ablation study on key components in LLM4TS.** Each ablation is conducted under both full- and few-shot learning with 10% training data. We report results averaged over prediction lengths $T_{out} \in \{96, 192, 336, 720\}$ for the ETTh1 dataset. The best results are in **bold**.

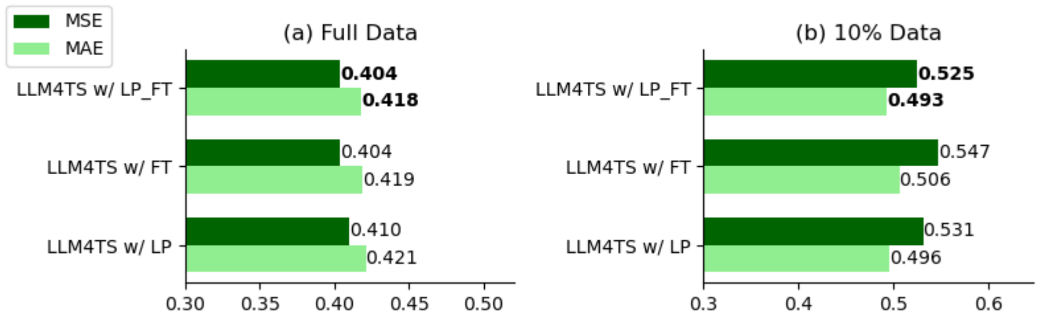


Figure 4: **Ablation study on training strategies in forecasting fine-tuning.** Each ablation is conducted under both full- and few-shot learning with 10% training data. We report results averaged over prediction lengths $T_{out} \in \{96, 192, 336, 720\}$ for the ETTh1 dataset. The best results are in **bold**.

LLMs is on few-shot learning, LLM4TS not only excels in this area but also outperforms all deep train-from-scratch methods, even when provided with full access to the datasets. This success is attributed to LLM4TS's two-stage fine-tuning process and its ability to incorporate multi-scale temporal information. In contrast, GPT4TS, despite leveraging GPT-2's pre-trained representation learning capabilities, does not surpass traditional train-from-scratch baselines in full-shot scenarios. This limitation becomes particularly evident when dealing with large amounts of training data, largely due to GPT4TS's lack of time-series alignment and absence of multi-scale temporal information integration, both of which are crucial for improving time-series forecasting performance. Interestingly, for the largest dataset (Traffic), PatchTST manages to outperform both LLM4TS and GPT4TS. This suggests that in full-shot settings, where sufficient data is available, traditional deep train-from-scratch models may occasionally surpass those using pre-trained LLMs.

We also compare the latest LLM-based methods for time-series forecasting: Time-LLM, TEMPO, and TEST. TEMPO, being a prompt-based method, is evaluated in a zero-shot setting and consistently reports zero-shot results in both few-shot and full-shot scenarios. Time-LLM emerges as the top-performing model, utilizing an innovative approach of reprogramming LLMs for time-series forecasting by converting time-series data into text-based prototypes and using prompts for guiding predictions. However, LLM4TS remains competitive, staying close to Time-LLM in performance. Notably, LLM4TS outperforms Time-LLM in few-shot scenarios, demonstrating its strong data efficiency and robustness when training data is limited.

Table 9: **Ablation study on the effectiveness of LLM's pre-trained weights**. Each ablation is conducted under few-shot learning with 10% and 5% training data. 'No Freeze' refers to the model utilizing LLM's pre-trained weights without freezing any layers during training, whereas 'No Pretrain' denotes the model not utilizing LLM's pre-trained weights, implying the model is trained from scratch. We report results averaged over prediction lengths $T_{out} \in \{96, 192, 336, 720\}$ for the Weather, ETTm1, and ETTm2 datasets. The best average results are in **bold**.

Methods		LLM4TS		GPT4TS		No Freeze		No Pretrain	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
10%	Weather	**0.235**	**0.270**	0.238	0.275	0.273	0.302	0.278	0.305
	ETTm1	**0.408**	**0.413**	0.464	0.441	0.546	0.484	0.473	0.446
	ETTm2	**0.276**	**0.324**	0.293	0.335	0.340	0.367	0.361	0.385
5%	Weather	**0.256**	**0.292**	0.264	0.302	0.284	0.312	0.298	0.324
	ETTm1	**0.413**	**0.417**	0.467	0.450	0.562	0.496	0.470	0.452
	ETTm2	**0.286**	**0.332**	0.308	0.347	0.327	0.362	0.413	0.411

B.7 Ablation Study

B.7.1 Key Components in LLM4TS

Figure 3 examines the impact of time-series alignment, multi-scale temporal encoding, and Parameter-Efficient Fine-Tuning (PEFT) on LLM4TS, evaluating both full- and few-shot scenarios on the ETTh1 dataset. A comparative analysis—with and without these components—demonstrates their individual significance in improving forecasting accuracy in both scenarios. LLM4TS achieves outstanding performance in few-shot learning, with an average 6.2% reduction in MSE when these components are integrated.

The experimental results reveal three key insights. First, there is a clear trend of greater MSE improvement as the prediction length increases. This suggests that the core components of LLM4TS become increasingly advantageous in scenarios requiring more advanced predictive capability, particularly for longer prediction lengths. Second, the gains are more pronounced in few-shot scenarios compared to full-shot ones when these components are added to LLM4TS. This underscores LLM4TS's strength as a data-efficient time-series forecaster, largely due to its built-in components. Third, among the two PEFT methods, LoRA outperforms Layer Normalization in both full-shot and few-shot scenarios. This consistent advantage highlights LoRA's effectiveness in boosting the model's overall performance.

B.7.2 Training Strategies in Forecasting Fine-Tuning

As outlined in Section 2.2, linear probing (LP) excels in out-of-distribution (OOD) scenarios, while full fine-tuning (FT) performs better in in-distribution (ID) cases. However, the LP-FT strategy, which combines both, can outperform FT and LP in both OOD and ID settings. Figure 4 shows that LP-FT improves performance in both full- and few-shot learning on the ETTh1 dataset, achieving an average MSE improvement of 0.7% in full-shot learning and 2.51% in few-shot learning. The relatively modest improvements in both scenarios can be attributed to the limited number of trainable parameters in LLM4TS's backbone model, even when applying FT, which reduces the gap between LP and FT. The findings further highlight that few-shot learning benefits more from LP-FT, likely due to its greater sensitivity to OOD issues. Additionally, as seen in the ablation study on LLM4TS's core components, longer prediction lengths tend to produce more significant gains in few-shot scenarios.

B.7.3 Effectiveness of LLM's Pre-Trained Weights

As discussed in Section A.1, most parameters in LLMs are kept fixed to maintain their data-independent representation learning capability. Table 9 shows that LLM4TS performs best when most of its parameters remain frozen, as observed on the Weather, ETTm1, and ETTm2 datasets. Specifically, LLM4TS achieves an average MSE improvement of 17.78% compared to the 'No Freeze' strategy, where pre-trained weights are used without freezing any layers during training. Additionally, when compared to the 'No Pretrain' approach—where the model is trained entirely

Table 10: **Training parameters**.

Model	Trainable Parameters	Total Parameters	Trainable Parameters Percentage
LLM4TS	3.4M	85M	4%
PatchTST	20M	20M	100%
FEDformer	33M	33M	100%

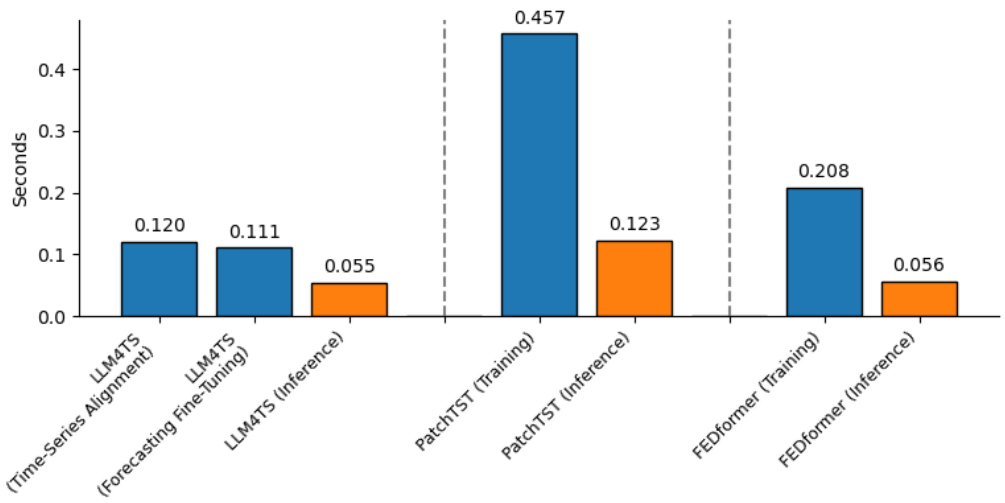


Figure 5: **Comparison of training and inference time (in seconds) for one batch.** We use the prediction length $T_{out} = 96$ for the ETTh2 dataset. The best results are in **bold**.

from scratch—LLM4TS shows an even larger average improvement of 18.28% in MSE. This highlights the critical importance of preserving the representation learning strengths that are inherent to pre-trained models, primarily due to the self-attention mechanisms in transformers, which facilitate data-independent operations.

B.8 Training and Inference Cost

Assessing the computational costs of LLM-based models is crucial for evaluating their feasibility in real-world applications. In this study, we compare LLM4TS with two other leading Transformer-based baselines, PatchTST and FEDformer. Details regarding the number of trainable and total parameters are provided in Table 10. LLM4TS stands out by keeping most of its pre-trained parameters frozen and utilizing two Parameter-Efficient Fine-Tuning (PEFT) methods: Layer Normalization Tuning and LoRA. As a result, only 4% of its parameters are trainable, significantly reducing the number of trainable parameters compared to its train-from-scratch counterparts.

The execution time for both training and inference for LLM4TS, PatchTST, and FEDformer was measured using an NVIDIA Tesla V100 GPU, and the results are shown in Figure 5. To ensure a fair comparison, we standardized the batch size to 128 and set the hidden dimensions to 768, consistent with GPT-2's specifications. The evaluation was conducted on a single batch, and for LLM4TS, the training time is reported for both stages, as the model undergoes a two-stage training process. LLM4TS demonstrated faster execution times during both the training and inference stages compared to the baselines. This improved efficiency can be attributed to LLM4TS's architecture, which retains most parameters as non-trainable, significantly reducing the computational load during both the training and inference phases.