LLM4TS: Two-Stage Fine-Tuning for Time-Series Forecasting with Pre-Trained LLMs

Ching Chang, Wen-Chih Peng, Tien-Fu Chen

National Yang Ming Chiao Tung University
blacksnail789521.cs10@nycu.edu.tw, {wcpeng, tfchen}@cs.nycu.edu.tw

Abstract

In this work, we leverage pre-trained Large Language Models (LLMs) to enhance time-series forecasting. Mirroring the growing interest in unifying models for Natural Language Processing and Computer Vision, we envision creating an analogous model for long-term time-series forecasting. Due to limited large-scale time-series data for building robust foundation models, our approach LLM4TS focuses on leveraging the strengths of pre-trained LLMs. By combining time-series patching with temporal encoding, we have enhanced the capability of LLMs to handle time-series data effectively. Inspired by the supervised fine-tuning in chatbot domains, we prioritize a two-stage fine-tuning process: first conducting supervised fine-tuning to orient the LLM towards time-series data, followed by task-specific downstream fine-tuning. Furthermore, to unlock the flexibility of pre-trained LLMs without extensive parameter adjustments, we adopt several Parameter-Efficient Fine-Tuning (PEFT) techniques. Drawing on these innovations, LLM4TS has yielded state-of-the-art results in long-term forecasting. Our model has also shown exceptional capabilities as both a robust representation learner and an effective few-shot learner, thanks to the knowledge transferred from the pre-trained LLM.

1 Introduction

Multivariate time-series data is prevalent across various domains. In many of these areas, forecasting stands out as one of the most crucial tasks, largely because it eliminates the need for manual labeling. There are numerous methods designed specifically for time-series forecasting (Zeng et al. 2023; Nie et al. 2023; Wu et al. 2021; Zhou et al. 2022, 2021), though some lean towards unsupervised representation learning (Yang and Hong 2022; Yue et al. 2022; Tonekaboni, Eytan, and Goldenberg 2021; Eldele et al. 2021). Regardless, the ultimate goal of attaining strong time-series forecasting performance lies in employing adept representation learners: first extracting meaningful representations from the time-series data and then using these learned representations for forecasting.

We have observed the emergence of foundation models in Natural Language Processing (NLP) (Radford et al. 2019; Brown et al. 2020) and Computer Vision (CV) (Dosovitskiy et al. 2021), with numerous successful applications leveraging these models for advanced representation learning. Given the limited availability of large-scale time-series data to train robust foundation models, we aim to investigate the potential of utilizing pre-trained Large Language Models (LLMs) as powerful representation learners. Notably, LLMs are recognized for their exceptional few-shot learning capabilities, making them ideal for real-world scenarios where collecting vast amounts of training data is a challenge.

To integrate LLMs with time-series data, we need to address two pivotal questions: 1. How can we input time-series data into LLMs? 2. How can we utilize pre-trained LLMs without distorting their inherent features?

How can we input time-series data into LLMs? To accommodate new data modalities within LLMs, it is essential to tokenize the data. Recent research (Zhang et al. 2023) highlights *patching* a promising tokenization approach, which is relevant for a variety of data types including images, audio, and time-series data. PatchTST (Nie et al. 2023) demonstrates that patching time-series data with channel-independence often yields improved results. Channel-independence entails treating multivariate time-series data as multiple univariate time-series data, and these are then processed by a single model. Furthermore, several Transformer-based time-series forecasting models suggest that integrating temporal information can enhance outcomes (Wen et al. 2022). As such, we introduce a novel approach that integrates temporal information while employing the techniques of patching and channel-independence.

How can we utilize pre-trained LLMs without distorting their inherent features? Building high-quality chatbots like InstructGPT (Ouyang et al. 2022) and ChatGPT requires strategic alignment of a pre-trained model with instruction-based data through *supervised fine-tuning*. This method ensures the model becomes familiarized with target data formats and characteristics. Motivated by this success, we introduce a two-stage fine-tuning approach. We start by guiding the LLM towards time-series data using supervised fine-tuning, then pivot to downstream fine-tuning geared towards time-series forecasting. Alongside this two-stage fine-tuning approach, there is a need to enhance the pre-trained LLMs' adaptability to new data modalities without distorting models' inherent features. We incorporate two Parameter-Efficient Fine-Tuning (PEFT) techniques, Layer Normalization Tuning (Lu et al. 2021) and LoRA (Hu et al. 2021), to optimize model flexibility without extensive parameter adjustments in the backbone model.

In summary, we introduce **LLM4TS**, a framework for time-series forecasting with pre-trained LLMs. Our main contributions can be listed as follows:

- **Integration of Time-Series with LLMs:** We utilize patching and channel-independence to tokenize time-series data, and we introduce a novel approach to integrate temporal information with patching.
- **Adaptable Fine-Tuning for LLMs:** We present a two-stage fine-tuning methodology: an initial *supervised fine-tuning* stage to align LLMs with time-series data characteristics, succeeded by a *downstream fine-tuning* stage dedicated to time-series forecasting.
- **Optimized Model Flexibility:** To ensure both robustness and adaptability in pre-trained LLMs for time-series data, we incorporate two PEFT techniques, Layer Normalization Tuning and LoRA.
- **Real-World Application Relevance:** Our methods cater to the challenges of limited time-series data availability, leveraging the few-shot learning capabilities of LLMs, to make our approach highly relevant for practical, real-world forecasting scenarios.

2 Related Work

2.1 In-Modality Knowledge Transfer

One standout benefit of using foundation models is their capability to transfer knowledge to downstream tasks. The transformation of LLMs into chatbots serves as a testament to this success. Prominent models like InstructGPT (Ouyang et al. 2022) and ChatGPT employ supervised fine-tuning to align with instruction-based data. While the benefits of fine-tuning are substantial, the computational burden of refining an entire model can be significant. This has led to the rise of PEFT as a popular technique to reduce costs (Lialin, Deshpande, and Rumshisky 2023). LLaMA-Adapter (Gao et al. 2023) achieves ChatGPT-level performance by fine-tuning a mere 0.02% of its parameters. In LLM4TS, we integrate supervised fine-tuning and PEFT to harness the potential of pretrained LLMs without the need for significant computational resources.

2.2 Cross-Modality Knowledge Transfer

Leveraging the prowess of foundation models in NLP and CV, we can extend their knowledge transfer capabilities across diverse data modalities. In NLP, numerous research has demonstrated the potential of pre-trained LLMs in tasks involving images (Lu et al. 2021), audio (Ghosal et al. 2023), and time-series data (Zhou et al. 2023). In CV, pre-trained ViT (Dosovitskiy et al. 2021) has been demonstrated to adapt successfully to tasks across 12 distinct modalities (Zhang et al. 2023). With LLM4TS, we utilize pretrained LLM expertise to address challenges in time-series data.

2.3 Long-term Time-Series Forecasting

Numerous efforts have been dedicated to employing Transformer models for long-term time-series forecasting (Nie et al. 2023; Zhou et al. 2021; Wu et al. 2021; Zhou et al. 2022). While Transformer-based models have gained traction, DLinear (Zeng et al. 2023) reveals that an *embarrassingly simple* single-layer linear model can surpass many of these sophisticated Transformer-based approaches. As we will illustrate, LLM4TS sets new benchmarks alongside these state-of-the-art approaches.

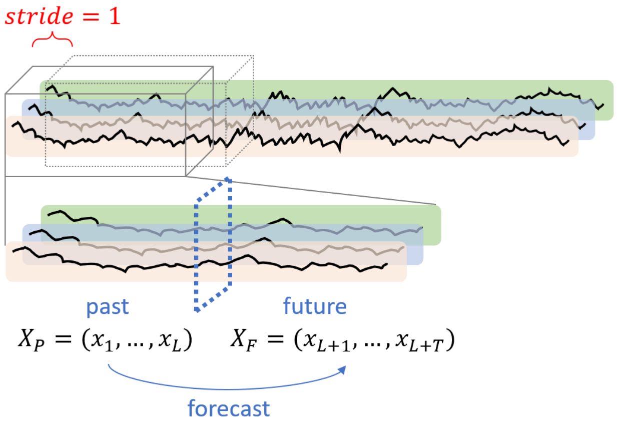

$stride = 1$
past
future
$X_P = (x_1, \ldots, x_L)$
$X_F = (x_{L+1}, \ldots, x_{L+T})$
forecast

Figure 1: Problem formulation for multivariate time-series forecasting.

2.4 Time-Series Representation Learning

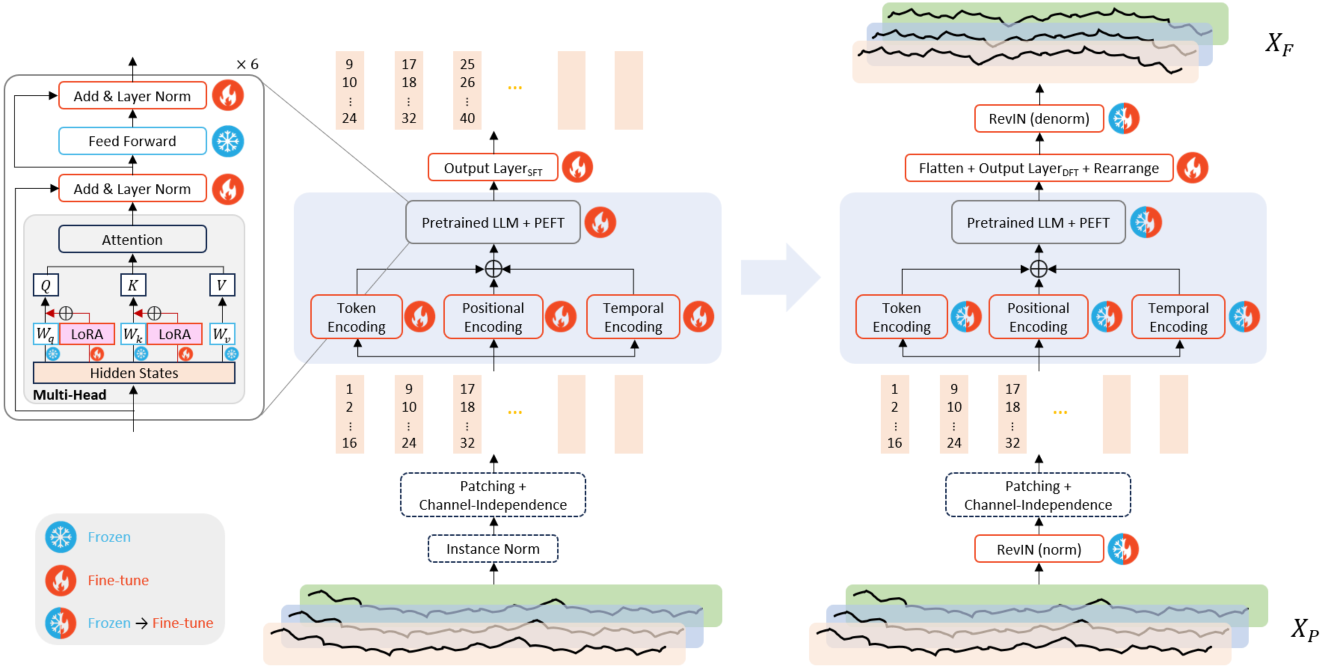
In the time-series domain, self-supervised learning emerges as a prominent approach to representation learning. While Transformers are widely recognized as prime candidates for foundation models (Bommasani et al. 2021), they don't consistently stand out as the preferred architecture in time-series self-supervised learning. Instead, CNN-based (Yue et al. 2022) or RNN-based (Tonekaboni, Eytan, and Goldenberg 2021) backbones continue to be favored for such tasks. LLM4TS seeks to reestablish the Transformer's dominance in this realm of self-supervised learning.

3 Problem Formulation

Given a complete and evenly-sampled multivariate time series, we use a sliding data window to extract sequential samples, as illustrated in Figure 1. This window moves with a stride of 1 and has a total length of $L + T$ — comprising past data $X_P = (x_1, \ldots, x_L)$ with a look-back window length L and future data $X_F = (x_{L+1}, \ldots, x_{L+T})$ with a prediction length T. For each time step t, x_t represents an M-dimensional vector. Our objective is to use the past data X_P to predict the future data X_F.

4 Method

In this section, we present our LLM4TS framework, leveraging the pre-trained GPT-2 (Radford et al. 2019) as the backbone model. In Section 4.1, we introduce the two-stage fine-tuning training strategy. Following that, Section 4.2 details the architecture, covering instance normalization, patching, channel-independence, three distinct encodings, the backbone model structure, and the output layer. All key components of our framework are illustrated in Figure 2.



(a) Supervised Fine-Tuning (SFT): Autoregressive

(b) Downstream Fine-Tuning (DFT): Forecasting

Figure 2: LLM4TS framework. (a) Supervised fine-tuning uses the autoregressive approach to align the backbone model with time-series data. (b) Downstream fine-tuning for time-series forecasting starts with linear probing, followed by full fine-tuning.

4.1 Two-Stage Fine-Tuning

Supervised Fine-tuning: Autoregressive Inspired by the achievements of chatbots, we have adopted supervised fine-tuning prior to downstream fine-tuning. It is well established that the training method for LLMs is algorithmically consistent in both pre-training on large corpus and subsequent fine-tuning to align with instruction-based data (Ouyang et al. 2022). Given our selection of GPT-2 (Radford et al. 2019) as the backbone model, which is a causal language model, we ensure that the supervised fine-tuning adopts the same *autoregressive* training methodology used during its pre-training phase. As illustrated in Figure 2(a), given an input sequence of patches such as 1^{st} patch, 2^{nd} patch, 3^{rd} patch, etc., the backbone model is expected to generate outputs corresponding to the 2^{nd} patch, 3^{rd} patch, 4^{th} patch, and so forth.

Downstream Fine-tuning: Forecasting After aligning the LLM with patched time-series data in the supervised fine-tuning stage, we transfer the trained weights of the backbone model, including those from the encoding layers, to the downstream fine-tuning stage for time-series forecasting. When adapting the backbone model for the downstream task, two primary strategies are available: full fine-tuning (where all model parameters are updated) and linear probing (where only the final linear layer is modified). Studies have shown that a sequential approach—initial linear probing followed by full fine-tuning (LP-FT)—consistently surpasses strategies that exclusively employ either method (Kumar et al. 2022). In our experiments, the initial half of the epochs is dedicated to linear probing, while the latter half focuses on full fine-tuning.

4.2 Architecture Details of LLM4TS Framework

While the inputs for both the supervised fine-tuning stage and the downstream fine-tuning stage are identical, not all components are shared between the two. Later, we will provide a comprehensive breakdown of the framework, tracing the path from the initial input to the final output across both stages.

Instance Normalization While z-score normalization is standard for time-series forecasting, Reversible Instance Normalization (RevIN) further boosts accuracy by applying additional batch-specific instance normalization and subsequent denormalization (Kim et al. 2021). Enhanced with a trainable affine transformation, RevIN has shown significant benefits in our downstream forecasting task.

Since RevIN is designed for the conventional (unpatched) time-series format and not for the patched variant, its direct application to the supervised fine-tuning stage is problematic for two reasons:

1. Denormalization is infeasible as outputs remain in the patched format rather than the unpatched format.
2. RevIN's trainable affine transformation is not appropriate for autoregressive models. Consider a scenario in which

we aim to use the 1^{st}, 2^{nd}, and 3^{rd} patches to predict the 2^{nd}, 3^{rd}, and 4^{th} patches. After assembling the necessary time-series data to create patches from the 1^{st} to the 4^{th} and applying instance normalization with a trainable affine transformation, the adjusted 2^{nd}, 3^{rd}, and 4^{th} patches are no longer suitable as ground truth due to alterations by the trainable transformation.

Given these issues, we employ standard instance normalization during the supervised fine-tuning stage.

Patching and Channel-Independence To adapt time-series data for Transformer-based models, we tokenize it using PatchTST's (Nie et al. 2023) method, which employs channel-independence and patching. Channel-independence treats multivariate time-series as multiple univariate time-series data, and these are then processed by a single model. While channel-mixing models aim to exploit cross-channel data directly, channel-independence frequently delivers superior performance by indirectly capturing cross-channel interactions through weight sharing. This is because channel-mixing often suffers from limited data and overfitting.

With channel-independence applied, the model perceives data as univariate time-series. The subsequent patching step groups adjacent time steps into a singular patch-based token. This approach broadens the input's historical scope without increasing token length, providing time and space advantages crucial for resource-intensive LLMs.

Three Encodings Given a series of tokens, it is necessary to apply token encoding to transform these tokens for compatibility with our GPT-2 backbone model. In conventional NLP practices, token encoding is typically achieved using a trainable lookup table to map tokens into a high-dimensional space. However, since our tokens are patched time-series data and represent vectors rather than scalars, we opt for a one-dimensional convolutional layer.

For positional encoding, we use the standard approach and employ a trainable lookup table to map patch locations.

When considering temporal encoding, numerous studies suggest the advantage of incorporating temporal information with Transformer-based models in time-series analysis (Wen et al. 2022). However, when dealing with patched time-series data, we face two challenges due to the need to aggregate multiple pieces of information into one unified representation:

1. Each patch encompasses multiple timestamps.
2. Each timestamp carries various temporal attributes like minute, hour, day of the week, date, and month.

In response to the first challenge of multiple timestamps within a patch, we designate the initial timestamp as its representative. To address the second challenge associated with diverse temporal attributes within a timestamp, we employ a trainable lookup table for each attribute, mapping it into a high-dimensional space, then summing them to produce a singular temporal embedding.

Finally, the token, positional, and temporal embeddings are summed to yield the final embedding, which is then fed into the pre-trained LLM.

Table 1: 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

Pre-Trained LLM and PEFT In our setup, GPT-2 (Radford et al. 2019) serves as the backbone model for our pre-trained LLM. To retain the model's foundational knowledge, we have frozen the majority of the parameters, particularly those associated with the multi-head attention and feed-forward layers within the Transformer block. Beyond the more cost-effective computational advantages, many studies indicate that retaining most parameters as non-trainable often yields better results than training a pre-trained LLM from scratch (Lu et al. 2021; Zhou et al. 2023).

For the remaining trainable parameters in the pre-trained LLM, we employ PEFT techniques as efficient approaches to selectively adjust or introduce a limited set of trainable parameters. We utilize the selection-based method, Layer Normalization Tuning (Lu et al. 2021), to adjust pre-existing parameters by making the affine transformation in layer normalization trainable. Concurrently, we employ LoRA (Low-Rank Adaptation) (Hu et al. 2021), a reparameterization-based method that leverages low-rank representations to adeptly refine the model without compromising its inherent expressiveness. With PEFT, only 1.5% of the model's total parameters are trainable.

Output Layer During the supervised fine-tuning stage, the output remains in the form of patched time-series data, essentially a sequence of tokens. To handle this, we employ a linear layer to modify the final dimension. In the downstream fine-tuning stage, the output layer transforms the patched time-series input into a general (unpatched) time-series format, requiring flattening before the linear layer and rearranging afterward. For the output layers in both stages, we incorporate dropout immediately after the linear transformation.

5 Experiments

Datasets In our LLM4TS framework evaluation, we use seven datasets: Weather, Traffic, Electricity, and four ETT sets (ETTh1, ETTh2, ETTm1, ETTm2). These widely-used multivariate time-series datasets (Wu et al. 2021) help us thoroughly test our framework. Detailed statistics for these datasets are provided in Table 1.

Baselines For long-term time-series forecasting, we select state-of-the-art models including GPT4TS (Zhou et al. 2023), DLinear (Zeng et al. 2023), PatchTST (Nie et al. 2023), FEDformer (Zhou et al. 2022), Autoformer (Wu et al. 2021), and Informer (Zhou et al. 2021). The same set of models is used for few-shot learning and ablation study. For self-supervised learning, we choose PatchTST, BTSF (Yang

Table 2: Long-term forecasting for multivariate time-series data. We use prediction lengths $T \in \{96, 192, 336, 720\}$ for all datasets. The best average results are in **bold**, while the second-best are <u>underlined</u>.

Methods		LLM4TS		GPT4TS		DLinear		PatchTST		FEDformer		Autoformer		Informer	
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	0.217	0.296	0.266	0.336	0.3	0.384
	192	0.191	0.238	0.204	0.248	0.22	0.282	0.194	0.241	0.276	0.336	0.307	0.367	0.598	0.434
	336	0.241	0.277	0.254	0.286	0.265	0.319	0.245	0.282	0.339	0.38	0.359	0.395	0.578	0.523
	720	0.313	0.329	0.326	0.337	0.333	0.362	0.314	0.334	0.403	0.428	0.419	0.428	1.059	0.741
	Avg.	**0.223**	**0.260**	0.237	0.271	0.249	0.300	<u>0.226</u>	<u>0.264</u>	0.309	0.360	0.338	0.382	0.634	0.521
ETTh1	96	0.371	0.394	0.376	0.397	0.375	0.399	0.37	0.399	0.376	0.419	0.449	0.459	0.865	0.713
	192	0.403	0.412	0.416	0.418	0.405	0.416	0.413	0.421	0.42	0.448	0.5	0.482	1.008	0.792
	336	0.42	0.422	0.442	0.433	0.439	0.443	0.422	0.436	0.459	0.465	0.521	0.496	1.107	0.809
	720	0.422	0.444	0.477	0.456	0.472	0.49	0.447	0.466	0.506	0.507	0.514	0.512	1.181	0.865
	Avg.	**0.404**	**0.418**	0.428	<u>0.426</u>	0.423	0.437	<u>0.413</u>	0.431	0.440	0.460	0.496	0.487	1.040	0.795
ETTh2	96	0.269	0.332	0.285	0.342	0.289	0.353	0.274	0.336	0.358	0.397	0.346	0.388	3.755	1.525
	192	0.328	0.377	0.354	0.389	0.383	0.418	0.339	0.379	0.429	0.439	0.456	0.452	5.602	1.931
	336	0.353	0.396	0.373	0.407	0.448	0.465	0.329	0.38	0.496	0.487	0.482	0.486	4.721	1.835
	720	0.383	0.425	0.406	0.441	0.605	0.551	0.379	0.422	0.463	0.474	0.515	0.511	3.647	1.625
	Avg.	<u>0.333</u>	<u>0.383</u>	0.355	0.395	0.431	0.447	**0.330**	**0.379**	0.437	0.449	0.450	0.459	4.431	1.729
ETTm1	96	0.285	0.343	0.292	0.346	0.299	0.343	0.29	0.342	0.379	0.419	0.505	0.475	0.672	0.571
	192	0.324	0.366	0.332	0.372	0.335	0.365	0.332	0.369	0.426	0.441	0.553	0.496	0.795	0.669
	336	0.353	0.385	0.366	0.394	0.369	0.386	0.366	0.392	0.445	0.459	0.621	0.537	1.212	0.871
	720	0.408	0.419	0.417	0.421	0.425	0.421	0.416	0.42	0.543	0.49	0.671	0.561	1.166	0.823
	Avg.	**0.343**	**0.378**	0.352	0.383	0.357	<u>0.379</u>	<u>0.351</u>	0.381	0.448	0.452	0.588	0.517	0.961	0.734
ETTm2	96	0.165	0.254	0.173	0.262	0.167	0.269	0.165	0.255	0.203	0.287	0.255	0.339	0.365	0.453
	192	0.22	0.292	0.229	0.301	0.224	0.303	0.22	0.292	0.269	0.328	0.281	0.34	0.533	0.563
	336	0.268	0.326	0.286	0.341	0.281	0.342	0.274	0.329	0.325	0.366	0.339	0.372	1.363	0.887
	720	0.35	0.38	0.378	0.401	0.397	0.421	0.362	0.385	0.421	0.415	0.433	0.432	3.379	1.338
	Avg.	**0.251**	**0.313**	0.267	0.326	0.267	0.334	<u>0.255</u>	<u>0.315</u>	0.305	0.349	0.327	0.371	1.410	0.810
ECL	96	0.128	0.223	0.139	0.238	0.14	0.237	0.129	0.222	0.193	0.308	0.201	0.317	0.274	0.368
	192	0.146	0.24	0.153	0.251	0.153	0.249	0.157	0.24	0.201	0.315	0.222	0.334	0.296	0.386
	336	0.163	0.258	0.169	0.266	0.169	0.267	0.163	0.259	0.214	0.329	0.231	0.338	0.3	0.394
	720	0.2	0.292	0.206	0.297	0.203	0.301	0.197	0.29	0.246	0.355	0.254	0.361	0.373	0.439
	Avg.	**0.159**	<u>0.253</u>	0.167	0.263	0.166	0.264	<u>0.162</u>	**0.253**	0.214	0.327	0.227	0.338	0.311	0.397
Traffic	96	0.372	0.259	0.388	0.282	0.41	0.282	0.36	0.249	0.587	0.366	0.613	0.388	0.719	0.391
	192	0.391	0.265	0.407	0.29	0.423	0.287	0.379	0.256	0.604	0.373	0.616	0.382	0.696	0.379
	336	0.405	0.275	0.412	0.294	0.436	0.296	0.392	0.264	0.621	0.383	0.622	0.337	0.777	0.42
	720	0.437	0.292	0.45	0.312	0.466	0.315	0.432	0.286	0.626	0.382	0.66	0.408	0.864	0.472
	Avg.	<u>0.401</u>	<u>0.273</u>	0.414	0.295	0.434	0.295	**0.391**	**0.264**	0.610	0.376	0.628	0.379	0.764	0.416

and Hong 2022), TS2Vec (Yue et al. 2022), TNC (Tonekaboni, Eytan, and Goldenberg 2021), and TS-TCC (Eldele et al. 2021). Consistent with prior research (Zeng et al. 2023; Nie et al. 2023; Zhou et al. 2023), we rely on Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics across all experiments.

Implementation Details For our experiments in long-term time-series forecasting, few-shot learning, and ablation study, we use the configurations from PatchTST (Nie et al. 2023) for a consistent comparison. Specifically, our look-back window length L is set to either 336 or 512, and we report the best results. With PatchTST's settings for patching, we set patch length $P = 16$ and stride $S = 8$. For self-supervised learning, the settings are slightly adjusted to $L = 512$, $P = 12$, and $S = 12$. Aligned with the GPT4TS configuration (Zhou et al. 2023), we utilize only the first 6 layers of the 12-layer GPT-2 base (Radford et al. 2019).

5.1 Long-Term Time-Series Forecasting

For all datasets, we present results using a consistent prediction length set $T \in \{96, 192, 336, 720\}$. Given our two-stage fine-tuning approach, models are labeled as *sft* models after **s**upervised **f**ine-**t**uning and as *dft* models after **d**ownstream **f**ine-**t**uning. On each dataset, a single *sft* model is utilized across all prediction lengths, while a distinct *dft* model is deployed for each prediction length. For *dft* models within the same dataset, we maintain consistent hyperparameters across different prediction lengths. The results are presented in Table 2. While the primary intent of using the pretrained LLM is for few-shot learning, LLM4TS still surpasses most baseline methods even when given access to the full dataset. Notably, LLM4TS claims the leading position in 9 of the 14 evaluations, covering 7 datasets and 2 metrics.

Table 3: Few-shot long-term forecasting using 10% of the training data. Results are averaged across four prediction lengths: $T \in \{96, 192, 336, 720\}$ for all datasets. The best average results are in **bold**, while the second-best are underlined.

Methods	LLM4TS		GPT4TS		DLinear		PatchTST		FEDformer		Autoformer		Informer	
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	0.284	0.324	0.300	0.342	0.597	0.495
ETTh1	**0.525**	**0.493**	0.590	0.525	0.691	0.600	0.633	0.542	0.639	0.561	0.702	0.596	1.199	0.809
ETTh2	**0.366**	**0.407**	0.397	0.421	0.605	0.538	0.415	0.431	0.466	0.475	0.488	0.499	3.872	1.513
ETTm1	**0.408**	**0.413**	0.464	0.441	0.411	0.429	0.501	0.466	0.722	0.605	0.802	0.628	1.192	0.821
ETTm2	**0.276**	**0.324**	0.293	0.335	0.316	0.368	0.296	0.343	0.463	0.488	1.342	0.930	3.370	1.440
ECL	**0.172**	**0.264**	0.176	0.269	0.180	0.280	0.180	0.273	0.346	0.427	0.431	0.478	1.195	0.891
Traffic	0.432	**0.303**	0.440	0.310	0.447	0.313	**0.430**	0.305	0.663	0.425	0.749	0.446	1.534	0.811

Table 4: Few-shot long-term forecasting using 5% of the training data. Results are averaged across four prediction lengths: $T \in \{96, 192, 336, 720\}$ for all datasets. The best average results are in **bold**, while the second-best are underlined.

Methods	LLM4TS		GPT4TS		DLinear		PatchTST		FEDformer		Autoformer		Informer	
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	0.310	0.353	0.311	0.354	0.584	0.528
ETTh1	**0.651**	**0.551**	0.682	0.560	0.750	0.611	0.695	0.569	0.659	0.562	0.722	0.599	1.225	0.817
ETTh2	**0.359**	**0.405**	0.401	0.434	0.828	0.616	0.439	0.448	0.441	0.457	0.470	0.489	3.923	1.654
ETTm1	0.413	**0.417**	0.472	0.450	**0.401**	**0.417**	0.527	0.476	0.731	0.593	0.796	0.621	1.163	0.792
ETTm2	**0.286**	**0.332**	0.308	0.346	0.399	0.426	0.315	0.353	0.381	0.405	0.389	0.434	3.659	1.490
ECL	**0.173**	**0.266**	0.179	0.273	0.177	0.276	0.181	0.277	0.267	0.353	0.346	0.405	1.281	0.930
Traffic	**0.418**	**0.295**	0.434	0.305	0.451	0.317	0.418	0.297	0.677	0.424	0.833	0.502	1.591	0.832

Table 5: Self-supervised learning evaluation in forecasting with linear probing. We use prediction lengths $T \in \{24, 48, 168, 336, 720\}$ for the ETTh1 dataset. The best average results are in **bold**, while the second-best are underlined.

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	0.61
	48	0.342	0.384	0.354	0.385	0.613	0.524	0.629	0.555	0.705	0.688	0.72	0.693
	168	0.401	0.415	0.419	0.424	0.64	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	0.79	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

5.2 Few-Shot Learning

Pretrained LLMs excel in few-shot learning due to their inherent knowledge (Brown et al. 2020), and we strive to replicate this success in long-term time-series forecasting. In our experiments, we maintain consistent splits for training, validation, and test sets in both standard learning (where the full training set is used) and few-shot learning. For few-shot scenarios, we limit the training data percentage. To showcase the effectiveness of few-shot learning with LLM4TS, we present results using 10% of the data in Table 3 and results using 5% of the data in Table 4. Both LLM4TS and GPT4TS (Zhou et al. 2023) consistently outperform most competitors across various datasets, thanks to the knowledge embedded in GPT-2. Notably, LLM4TS outperforms GPT4TS on every dataset without exception.

5.3 Self-Supervised Learning

Given that autoregressive is a prevalent self-supervised learning technique, we aim to assess the representation learning capabilities of the *sft* model. To evaluate the effectiveness of self-supervised learning, we use linear probing after completing this stage. With the backbone model's parameters fixed, obtaining strong performance on the downstream task requires highly expressive learned representations. We use the ETTh1 dataset for long-term time-series forecasting to evaluate the effectiveness of learned representation. As showcased in Table 5, LLM4TS outperforms the competition in this assessment.

5.4 Ablation Study

Supervised Fine-Tuning, Temporal Encoding, and PEFT In Table 6, we examine the impact of three key components: supervised fine-tuning, temporal encoding, and PEFT on forecasting. Both standard and few-shot learning approaches are evaluated on the ETTh1 dataset. A comparative analysis of results—with and without these three components—highlights their individual importance in enhancing forecasting accuracy in both full-data and limited-data scenarios. Notably, LLM4TS delivers exceptional performance

Table 6: Ablation study of supervised fine-tuning, temporal encoding, and PEFT in LLM4TS. Each ablation is conducted under both standard learning and few-shot learning with 10% training data. IMP. denotes the average improvement achieved by incorporating one of LLM4TS's core components. We use prediction lengths $T \in \{96, 192, 336, 720\}$ for the ETTh1 dataset. The best average results are in **bold**.

Methods		IMP.	LLM4TS		w/o Supervised Fine-Tuning		w/o Temporal Encoding		w/o PEFT	
Metric		MSE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1 (full)	96	0.89%	0.371	0.394	0.372	0.395	0.378	0.397	0.373	0.393
	192	1.14%	0.403	0.412	0.404	0.411	0.411	0.416	0.408	0.412
	336	1.16%	0.42	0.422	0.422	0.423	0.433	0.43	0.42	0.421
	720	2.21%	0.422	0.444	0.424	0.448	0.442	0.457	0.429	0.45
	Avg.	1.37%	**0.404**	**0.418**	0.406	0.419	0.416	0.425	0.408	0.419
ETTh1 (10%)	96	1.80%	0.417	0.432	0.43	0.438	0.422	0.434	0.422	0.433
	192	1.01%	0.469	0.468	0.488	0.474	0.463	0.465	0.471	0.462
	336	4.03%	0.505	0.499	0.538	0.506	0.516	0.508	0.525	0.504
	720	11.89%	0.708	0.572	0.762	0.589	0.714	0.584	0.98	0.672
	Avg.	6.20%	**0.525**	**0.493**	0.555	0.502	0.529	0.498	0.600	0.518

Table 7: Ablation study of training strategies during downstream fine-tuning. Each ablation is conducted under both standard learning and few-shot learning with 10% training data. IMP. denotes the average improvement achieved by incorporating one of LLM4TS's core components. We use prediction lengths $T \in \{96, 192, 336, 720\}$ for the ETTh1 dataset. The best average results are in **bold**.

Methods		IMP.	LLMTS w/ LP_FT		LLMTS w/ FT		LLMTS w/ LP	
Metric		MSE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1 (full)	96	0.80%	0.371	0.394	0.371	0.394	0.377	0.398
	192	0.98%	0.403	0.412	0.404	0.413	0.41	0.416
	336	0.70%	0.42	0.422	0.42	0.423	0.426	0.424
	720	0.35%	0.422	0.444	0.42	0.444	0.427	0.447
	Avg.	0.70%	**0.404**	**0.418**	0.404	0.419	0.410	0.421
ETTh1 (10%)	96	0.12%	0.421	0.435	0.423	0.436	0.42	0.433
	192	2.52%	0.454	0.457	0.477	0.474	0.455	0.454
	336	2.36%	0.515	0.504	0.545	0.524	0.511	0.507
	720	3.92%	0.711	0.574	0.743	0.589	0.737	0.589
	Avg.	2.51%	**0.525**	**0.493**	0.547	0.506	0.531	0.496

in few-shot learning, averaging a 6.2% reduction in MSE with each incorporation of these components.

Training Strategies in Downstream Fine-Tuning When pre-trained models are robust and possess embedded knowledge, fully fine-tuning (FT) them on a downstream task might distort the learned representation, especially when a distribution shift exists between the training and test sets. A straightforward solution is to first apply linear probing (LP), followed by full fine-tuning. Empirically, the combined approach (LP-FT) consistently surpasses either LP or FT alone, regardless of the magnitude of the distribution shift (Kumar et al. 2022). For our experiments, we employ linear probing for the initial half of the training epochs and transition to full fine-tuning for the latter half. This method enhances the performance of the *dft* models in both standard and few-shot learning, as illustrated in Table 7. We observe that few-shot learning appears to derive greater benefit from the LP-FT method compared to standard learning, likely because few-shot scenarios are more susceptible to significant distribution shifts. Although the benefits of using LP-FT in both scenarios might seem subtle, there are no drawbacks to its adoption. The subtle improvements observed with LP-FT can be linked to the fact that only a small portion of the parameters in the LLM4TS's backbone model are trainable, which narrows the distinction between LP and FT.

6 Conclusion

In this paper, we introduce LLM4TS, a framework for time-series forecasting with pre-trained LLMs. We employ patching and channel-independence techniques while integrating temporal information. Our two-stage fine-tuning methodology first aligns LLMs with time-series data characteristics and then focuses on time-series forecasting tasks. To enhance the pre-trained LLMs' adaptability to this new modality of time-series data without distorting inherent features, we incorporate Layer Normalization Tuning and LoRA to enhance the model's robustness and versatility. These PEFT strategies further optimize computational efficiency during fine-tuning. In our evaluations, LLM4TS not only sets new benchmarks in long-term forecasting and representation learning but also excels in few-shot learning, making it a top choice for real-world scenarios with limited data availability.

References

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; Brynjolfsson, E.; Buch, S.; Card, D.; Castellon, R.; Chatterji, N. S.; Chen, A. S.; Creel, K. A.; Davis, J.; Demszky, D.; Donahue, C.; Doumbouya, M.; Durmus, E.; Ermon, S.; Etchemendy, J.; Ethayarajh, K.; Fei-Fei, L.; Finn, C.; Gale, T.; Gillespie, L. E.; Goel, K.; Goodman, N. D.; Grossman, S.; Guha, N.; Hashimoto, T.; Henderson, P.; Hewitt, J.; Ho, D. E.; Hong, J.; Hsu, K.; Huang, J.; Icard, T. F.; Jain, S.; Jurafsky, D.; Kalluri, P.; Karamcheti, S.; Keeling, G.; Khani, F.; Khattab, O.; Koh, P. W.; Krass, M. S.; Krishna, R.; Kuditipudi, R.; Kumar, A.; Ladhak, F.; Lee, M.; Lee, T.; Leskovec, J.; Levent, I.; Li, X. L.; Li, X.; Ma, T.; Malik, A.; Manning, C. D.; Mirchandani, S.; Mitchell, E.; Munyikwa, Z.; Nair, S.; Narayan, A.; Narayanan, D.; Newman, B.; Nie, A.; Niebles, J. C.; Nilforoshan, H.; Nyarko, J. F.; Ogut, G.; Orr, L. J.; Papadimitriou, I.; Park, J. S.; Piech, C.; Portelance, E.; Potts, C.; Raghunathan, A.; Reich, R.; Ren, H.; Rong, F.; Roohani, Y. H.; Ruiz, C.; Ryan, J.; R'e, C.; Sadigh, D.; Sagawa, S.; Santhanam, K.; Shih, A.; Srinivasan, K. P.; Tamkin, A.; Taori, R.; Thomas, A. W.; Tramèr, F.; Wang, R. E.; Wang, W.; Wu, B.; Wu, J.; Wu, Y.; Xie, S. M.; Yasunaga, M.; You, J.; Zaharia, M. A.; Zhang, M.; Zhang, T.; Zhang, X.; Zhang, Y.; Zheng, L.; Zhou, K.; and Liang, P. 2021. On the Opportunities and Risks of Foundation Models. *ArXiv*, abs/2108.07258.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*. OpenReview.net.

Eldele, E.; Ragab, M.; Chen, Z.; Wu, M.; Kwoh, C.; Li, X.; and Guan, C. 2021. Time-Series Representation Learning via Temporal and Contextual Contrasting. In *International Joint Conference on Artificial Intelligence*.

Gao, P.; Han, J.; Zhang, R.; Lin, Z.; Geng, S.; Zhou, A.; Zhang, W.; Lu, P.; He, C.; Yue, X.; Li, H.; and Qiao, Y. J. 2023. LLaMA-Adapter V2: Parameter-Efficient Visual Instruction Model. *ArXiv*, abs/2304.15010.

Ghosal, D.; Majumder, N.; Mehrish, A.; and Poria, S. 2023. Text-to-Audio Generation using Instruction-Tuned LLM and Latent Diffusion Model. *ArXiv*, abs/2304.13731.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

Kumar, A.; Raghunathan, A.; Jones, R.; Ma, T.; and Liang, P. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*.

Lialin, V.; Deshpande, V.; and Rumshisky, A. 2023. Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. *ArXiv*, abs/2303.15647.

Lu, K.; Grover, A.; Abbeel, P.; and Mordatch, I. 2021. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 1.

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

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744.

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

Tonekaboni, S.; Eytan, D.; and Goldenberg, A. 2021. Unsupervised Representation Learning for Time Series with Temporal Neighborhood Coding. In *ICLR*. OpenReview.net.

Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; and Sun, L. 2022. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34: 22419–22430.

Yang, L.; and Hong, S. 2022. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *International Conference on Machine Learning*, 25038–25054. PMLR.

Yue, Z.; Wang, Y.; Duan, J.; Yang, T.; Huang, C.; Tong, Y.; and Xu, B. 2022. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8980–8987.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11121–11128.

Zhang, Y.; Gong, K.; Zhang, K.; Li, H.; Qiao, Y.; Ouyang, W.; and Yue, X. 2023. Meta-Transformer: A Unified Framework for Multimodal Learning. *arXiv preprint arXiv:2307.10802*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, 27268–27286. PMLR.

Zhou, T.; Niu, P.; Wang, X.; Sun, L.; and Jin, R. 2023. One Fits All: Power General Time Series Analysis by Pretrained LM. *arXiv preprint arXiv:2302.11939*.