Generalized Prompt Tuning: How to Use a Frozen Pre-Trained Univariate Time Series Foundation Model for Multivariate Time Series Prediction

Mingzhu Liu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
mingzhul@andrew.cmu.edu

Angela H. Chen
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
angelachen@cmu.edu

George H. Chen
Heinz College
Carnegie Mellon University
Pittsburgh, PA 15213
georgechen@cmu.edu

Abstract

Time series foundation models are pre-trained on large datasets and are able to achieve state-of-the-art performance in diverse tasks. However, we observe that currently, the majority of time series foundation models either are univariate in nature, or assume channel independence, meaning that they handle multivariate time series but do not model how the different variables relate. In this paper, we propose a prompt-tuning-inspired fine-tuning technique, Generalized Prompt Tuning (Gen-P-Tuning), that enables us to adapt an existing univariate time series foundation model (treated as frozen) to handle multivariate time series prediction. Our approach provides a way to combine information across channels (variables) of multivariate time series. We demonstrate the effectiveness of our fine-tuning approach against various baselines on 8 classification and 4 forecasting datasets.

Our code is available at:
https://github.com/Ilovecodingforever/Gen-P-Tuning

1 Introduction

With the rapid development of large language models (LLMs) recently, there has been a surge of interest in developing similar sorts of foundation models for time series analysis (e.g., Das et al. 2023, Gruver et al. 2024, Goswami et al. 2024). They are trained on a variety of time series data, with the idea that time series across disciplines likely share similar patterns. However, a major limitation of most time series foundation models that have been developed is that they are univariate [Ye et al., 2024].

In this paper, our main contribution is to show how to adapt existing *unviariate* time series foundation models to handle *multivariate* time series prediction, specifically for both classification and forecasting. We treat the univariate time series foundation model as frozen, and how we adapt it to handle multivariate time series prediction is as a form of *parameter-efficient fine-tuning* (PEFT). PEFT methods have become popular recently for adapting LLMs to handling various datasets (e.g., Hu et al. 2022). Our proposed PEFT method is a generalization of an existing PEFT method called *prompt tuning*, of which we specifically generalize the variant called *P-tuning v2* by Liu et al. [2022]. As such, we call our proposed PEFT method *Generalized Prompt Tuning (Gen-P-Tuning)*.

We show that Gen-P-Tuning is competitive in practice against various fine-tuning baselines in experiments on classification and forecasting. To the best of our knowledge, our paper is also the first to benchmark various fine-tuning strategies on time series foundation models.

Bibliographic note A longer version of this paper specialized to healthcare data has been published at *Machine Learning for Health (ML4H)* [Liu et al., 2024]. In this short workshop paper, we apply the same method as in our ML4H paper to a wider range of datasets.

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

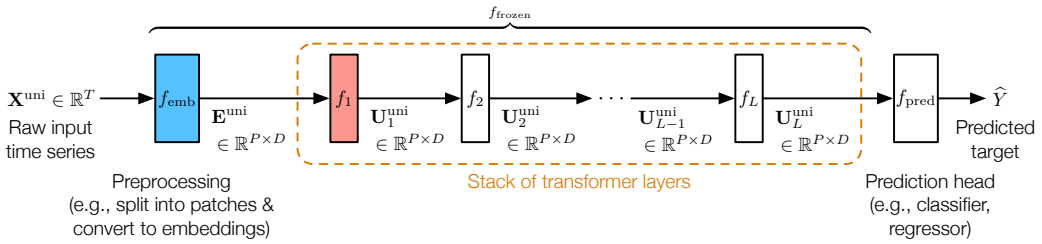

Figure 1: Univariate time series foundation model. The predicted target's dimensions depend on the prediction task ($\widehat{Y} \in [0,1]^M$ for M-way classification, and $\widehat{Y} \in \mathbb{R}^H$ for forecasting H time steps).

2 Background

2.1 Basic Notation

We denote a single multivariate input time series as $\mathbf{X} \in \mathbb{R}^{C \times T}$, where C is the number of channels/variables, and T is the number of time steps. The multivariate case corresponds to $C \geq 2$, whereas in the univariate case ($C = 1$), we represent the time series as a 1D array of length T. We denote the ground truth prediction target as $\mathbf{Y}$. If the prediction task is classification with $M \geq 2$ classes, then $\mathbf{Y} \in [0,1]^M$. If the prediction task is forecasting, then $\mathbf{Y} \in \mathbb{R}^{C \times H}$, where H is the time horizon.

2.2 Univariate Time Series Foundation Models

We now state the general form of the univariate foundation models that our proposed fine-tuning strategy can adapt into multivariate time series predictors. The structure is shown in Figure 1. A more detailed discussion on the structure is in Section 2 of the longer version of this paper [Liu et al., 2024].

For an input *univariate* time series $\mathbf{X}^{\text{uni}} \in \mathbb{R}^T$ where T is the number of time steps , the foundation model first applies a preprocessing function f_{emb} to $\mathbf{X}^{\text{uni}}$ to produce a preprocessed array

$$\mathbf{E}^{\text{uni}} = f_{\text{emb}}(\mathbf{X}^{\text{uni}}) \in \mathbb{R}^{P \times D}, \tag{1}$$

where P could be different from T due to patching [Nie et al., 2023] and D is the embedding dimension. The transformer backbone then takes in $\mathbf{E}^{\text{uni}}$ to produce intermediate embeddings $\mathbf{U}_\ell^{\text{uni}} \in \mathbb{R}^{P \times D}$ with transformer layers f_ℓ, where $\ell = 1, \ldots, L$. Finally, a prediction layer ("head") f_{pred} is applied to the last transformer layer's output $\mathbf{U}_L^{\text{uni}}$ to produce the final prediction:

$$\widehat{\mathbf{Y}} = f_{\text{pred}}(\mathbf{U}_L^{\text{uni}}),$$

In summary, the full univariate foundation model could be represented as the function

$$f \triangleq f_{\text{pred}} \circ \underbrace{(f_L \circ f_{L-1} \circ \cdots \circ f_1 \circ f_{\text{emb}})}_{\triangleq f_{\text{frozen}}}, \tag{2}$$

so that $\widehat{\mathbf{Y}} = f(\mathbf{X}^{\text{uni}}) = f_{\text{pred}}\big(f_{\text{frozen}}(\mathbf{X}^{\text{uni}})\big)$.

2.3 Handling Multivariate Time Series With Channel Independence

Given an input multivariate time series $\mathbf{X} \in \mathbb{R}^{C \times T}$, channel-independent models separate it out into its different channels' time series $\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \ldots, \mathbf{X}_{(C)} \in \mathbb{R}^T$. Then we can obtain the final transformer layer's output across the different channels:

$$\mathbf{U}_L^{(c)} \triangleq f_{\text{frozen}}(\mathbf{X}_{(c)}) \qquad \text{for } c = 1, \ldots, C. \tag{3}$$

Finally, we train a prediction head (such as a multilayer perceptron) that takes in $\mathbf{U}_L^{(1)}, \ldots, \mathbf{U}_L^{(C)}$ as the inputs (or the average of these across channels as a single input) and outputs the predicted target. Importantly, the univariate foundation model itself does not combine information across channels. Instead, only the prediction head does.

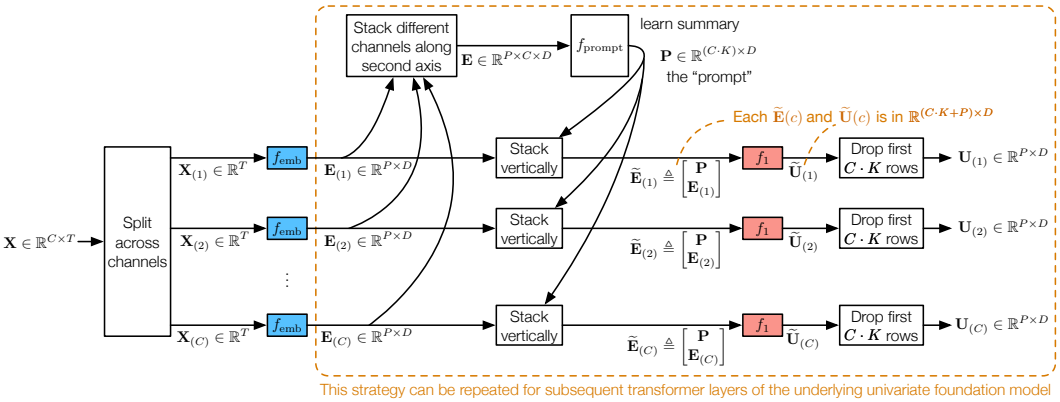

Figure 2: Overview of how the prompt $\mathbf{P}$ is added to the backbone. Note that the prompt is learned using a Prompt Module f_{prompt} that summarizes information across all channels. The prompt is then attached as a prefix to each channel's embedding $\mathbf{E}_{(1)}, \ldots, \mathbf{E}_{(C)}$. Blue and red boxes are the same as the ones in Figure 1.

3 Method

We now explain how our Gen-P-Tuning method works for adapting a pre-trained univariate time series foundation model for multivariate time series prediction. We treat the univariate foundation model as frozen, and our fine-tuning approach introduces trainable elements.

Overview We only provide a high-level overview of our Gen-P-Tuning approach here; for details, see Section 3.1 of the longer version of this paper [Liu et al., 2024]. Similar to P-tuning v2, Gen-P-Tuning trains a prompt for each transformer layer. It suffices to explain how Gen-P-Tuning works when there is a single transformer layer, as shown in Figure 2. Formally, given a frozen channel-independent model, for each channel's embedding $\mathbf{E}_{(1)}, \ldots, \mathbf{E}_{(C)} \in \mathbb{R}^{P \times D}$ (outputs of the blue boxes in Figure 2; each blue box is the same as the one from Figure 1), a prompt $\mathbf{P} \in \mathbb{R}^{(C \cdot K) \times D}$ is trained using the Prompt Module f_{prompt} (defined momentarily), where hyperparameter K controls the prompt size. We then attach the prompt $\mathbf{P}$ to each channel's embedding $\mathbf{E}_{(c)}$ (for $c = 1, \ldots, C$) to obtain

$$\widetilde{\mathbf{E}}_{(c)} \triangleq \begin{bmatrix} \mathbf{P} \\ \mathbf{E}_{(c)} \end{bmatrix} \in \mathbb{R}^{(C \cdot K + P) \times D},$$

which we then feed to the univariate foundation model's transformer layer f_1 (red boxes in Figure 2; this is the same red box as in Figure 1). We design the prompt such that it depends on information contained in all channels.

The Prompt Module The Prompt Module f_{prompt} summarizes the different preprocessed time series across channels (a total of $P \times C \times D$ numbers) into a single array $\mathbf{P} \in \mathbb{R}^{(C \cdot K) \times D}$ that notably does not depend on the number of patches P. In particular, f_{prompt} needs to accommodate the possibility that different input time series even within the same dataset could have different numbers of patches P (and across different datasets, the number of channels C could vary). Note that the Prompt Module consists of all trainable elements of Gen-P-Tuning aside from the final prediction head.

There are many ways to define f_{prompt}. We specifically define it to do the following steps:

1. We apply a transformer module to $\mathbf{E} \in \mathbb{R}^{P \times C \times D}$ by treating the P different patches as if they are different data points (so that each "data point" is in $\mathbb{R}^{C \times D}$, where C is treated as the "time steps" by the transformer module). The output per patch is in $\mathbb{R}^{C \times D}$, and we stack these outputs into a single array $\mathbf{P}' \in \mathbb{R}^{P \times C \times D}$.
2. We use a transformer to map $\mathbf{P}'$ from $\mathbb{R}^{P \times C \times D}$ to $\mathbb{R}^{P \times C \times (K \cdot D)}$, followed by a max pooling operation to map it to $\mathbb{R}^{C \times (K \cdot D)}$, and finally a reshape operation to map it to $\mathbb{R}^{(K \cdot C) \times D}$. Note that there are other ways of aggregating across the P dimension (to make the output of the Prompt Module not depend on the number of patches) such as recurrent neural networks and multi-layer perceptrons.

Table 1: Average test accuracy on classification tasks. For each univariate foundation model, per column we bold whichever score is the best and underline the second-best score.

	Full	LoRA	Linear Probing	Prompt Tuning	Gen-P-Tuning
MOMENT	0.732 ± 0.217	$\underline{0.763} \pm 0.206$	$\mathbf{0.769} \pm 0.204$	0.756 ± 0.205	0.761 ± 0.207
GPT4TS	$\underline{0.741} \pm 0.213$	$\mathbf{0.742} \pm 0.204$	$\mathbf{0.742} \pm 0.202$	0.609 ± 0.221	0.590 ± 0.243

Table 2: Average test MSE and MAE on forecasting tasks.This table uses the same formatting as Table 1 (in terms of what bolding and underlining mean).

		Full	LoRA	Linear Probing	Prompt Tuning	Gen-P-Tuning
MOMENT	MSE	1.211 ± 1.161	1.121 ± 1.152	3.252 ± 1.965	$\mathbf{1.071} \pm 1.068$	$\underline{1.118} \pm 1.137$
	MAE	0.698 ± 0.342	$\mathbf{0.631} \pm 0.318$	1.208 ± 0.302	$\underline{0.641} \pm 0.325$	0.647 ± 0.323
GPT4TS	MSE	1.612 ± 1.313	1.546 ± 1.336	3.242 ± 1.757	$\mathbf{1.478} \pm 1.255$	$\underline{1.482} \pm 1.251$
	MAE	0.807 ± 0.375	$\mathbf{0.751} \pm 0.372$	1.212 ± 0.270	0.763 ± 0.380	$\underline{0.760} \pm 0.366$

4 Experiments

Our experiments aim to show how different fine-tuning strategies for adapting univariate time series foundation models to multivariate time series classification and forecasting work in practice. To this end, we specifically consider univariate time series foundation models, MOMENT [Goswami et al., 2024] and GPT4TS [Zhou et al., 2023] that support both classification and forecasting, and both are special cases of the formulation we presented in Section 2.2. We use 8 datasets for classification and 4 datasets for forecasting. Details on datasets and implementation details are in Appendix A.

Baselines We compare Gen-P-Tuning to the following fine-tuning baselines:

1. Full fine-tuning, where all parameters are updated.
2. Low-Rank Adaptation (LoRA) [Hu et al., 2022].
3. Linear probing. For MOMENT, the only learnable part is now the prediction head. For GPT4TS, this includes the (non-pre-trained) input embedding layer and the prediction head.
4. P-tuning v2 [Liu et al., 2022] (referred to simply as "Prompt Tuning" in our tables later), where $\mathbf{P}$ is an array of trainable parameters.

The last two baselines are actually special cases of Gen-P-Tuning. Linear probing corresponds to a degenerate case where prompt size hyperparameter K is set to be 0 so that the prompt $\mathbf{P}$ has 0 rows; the resulting method is precisely the channel-independent strategy in Section 2.3. P-tuning v2 results from the Prompt Module f_{prompt} not actually depending on any inputs and instead just outputting a table of numbers, all of which are directly treated as neural network parameters.

Results We report average accuracy across the 8 classification datasets in Table 1, and the average MSE and MAE across 4 forecasting datasets in Table 2. Full results are included in Appendix B. Note that in these tables, "Gen-P-Tuning" refers to our proposed strategy used with a nontrivial Prompt Module (so that it is not reducing to the special case of either linear probing or P-tuning v2).

No method always performs the best. Gen-P-Tuning (with a nontrivial Prompt Module) is often among the best-performing ones. Linear probing sometimes performs very well, which suggests that a channel-independent strategy is sometimes sufficient. Full fine-tuning sometimes does not perform well, especially for forecasting experiments. This might be a sign of catastrophic forgetting [Goodfellow et al., 2014], which is commonly observed in low-data regimes with deep networks. There has not been previous work on catastrophic forgetting of time series foundation models, but it has been observed in models such as LSTMs [Schak and Gepperth, 2019].

In practice, note that one could tune Gen-P-Tuning (whether to have the prompt size be 0 to recover linear probing, use a trivial Prompt Module to obtain P-tuning v2, or use various nontrivial Prompt Modules) based on a validation set accuracy metric.

5 Discussion

A limitation of our work is that most of the datasets that we used were in MOMENT's pre-training dataset, although we ensured that the test portion was not part of the pre-training dataset. A direction that we have not explored is interpreting how the learned Prompt Module combines information across channels. While attention weights of the Prompt Module could be visualized, there is debate on whether attention weights are interpretable (e.g., Serrano and Smith 2019).

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their helpful feedback. G. H. Chen is supported by NSF CAREER award #2047981.

References

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*, 2023.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *International Conference on Learning Representations*, 2014.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G. Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems*, 2024.

Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(96), 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li wei H. Lehman, Mengling Feng, Mohammad Mahdi Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3, 2016.

Mingzhu Liu, Angela H. Chen, and George H. Chen. Generalized prompt tuning: Adapting frozen univariate time series foundation models for multivariate healthcare time series. In *Machine Learning for Health (ML4H)*, 2024.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. In *Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2022.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

Monika Schak and Alexander Gepperth. A study on catastrophic forgetting in deep LSTM networks. In *Artificial Neural Networks and Machine Learning – ICANN 2019: Deep Learning*, pages 714–728. Springer, 2019.

Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Annual Meeting of the Association for Computational Linguistics*, 2019.

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

Jiexia Ye, Weiqi Zhang, Ke Yi, Yongzi Yu, Ziyue Li, Jia Li, and Fugee Tsung. A survey of time series foundation models: Generalizing time series representation with large language model. *arXiv preprint arXiv:2405.02358*, 2024.

Tian Zhou, Peisong Niu, xue wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained lm. In *Advances in Neural Information Processing Systems*, 2023.

A Experimental Setup Details

Datasets We use 4 datasets (ETTh1, ETTh2, Exchange, ILI) from the dataset used in the Autoformer paper [Wu et al., 2021] for forecasting tasks and 6 datasets (EthanolConcentration, JapaneseVowels, SelfRegulationSCP1, SelfRegulationSCP2, SpokenArabicDigits, UWaveGestureLibrary) from the UEA archive [Bagnall et al., 2018] for classification tasks. These datasets are split into 60% training, 10% validation, and 30% testing. The forecasting horizon is 96 hours for ETTh1 and ETTh2, 96 days for Exchange, and 60 weeks for ILI. We forecast all 8 variables in Exchange, and 7 variables in ETTh1, ETTh2, and ILI. We point out that Goswami et al. [2024] also presented forecasting results on these datasets but they report experimental results only on forecasting one of the variables, so the numbers they get are not directly comparable to ours.

We also performed two classification experiments on MIMIC-III [Johnson et al., 2016], which is a publicly available electronic health records dataset collected from patients in the intensive care units of the Beth Israel Deaconess Medical Center from 2001 to 2012. We follow the benchmark proposed by Harutyunyan et al. [2019]. Specifically, we focus on two tasks. The first is a binary classification task of predicting in-hospital mortality based on the first 48 hours of an ICU stay (referred to as "MIMIC Mortality" in our tables later). The second is a multi-class, multi-label classification task, where we classify which of 25 acute care conditions occurred in an ICU stay ("MIMIC Phenotyping"). For these two MIMIC classification tasks, to simulate a resource-constrained environment, we use only 1000 randomly sampled patients, 60% of the data for training, 10% for validation, and 30% for testing. In the original benchmark, 17 clinical variables are included. We also include as an additional variable the number of hours since the time of admission to the ICU. Instead of one-hot encoding as in the benchmark, we encode categorical variables as ordinal values. The time series are irregularly sampled, and missing values are imputed using forward filling when possible or using the normal values suggested by the benchmark otherwise.

Note that the longer version of this paper [Liu et al., 2024] (written prior to this paper) uses a subset of the datasets we mention here (ILI forecasting and the two MIMIC classification tasks).

Implementation details Experiments were run with NVIDIA RTX A6000, Python 3.11.5, Pytorch 2.4.0, Huggingface-hub 0.24.0, and MOMENT-1-large. Experiments were run with the following hyperparameters (we did not tune hyparparameters):

- Number of epochs: 10
- Scheduler: OneCycleLR
- Optimizer: AdamW
- Learning rate: 5×10^{-5}
- Max learning rate: 0.01
- Weight decay: 0.05
- Loss function: binary cross-entropy for classification, MSE for forecasting
- Prompt size K: 4 for MIMIC experiments, 16 for others
- LoRA hyperparameters:
  - Attention dimension: 1 (attention dimension was chosen to make the number of trainable parameters of LoRA close to that of Gen-P-Tuning)
  - Alpha: 16
  - Dropout: 0.1

B Full Results

Results on classification datasets are in Table 3 (excludes MIMIC experiments), Table 4 (MIMIC Mortality), and Table 5 (MIMIC Phenotyping). Note that for MIMIC experiments, we additionally use the area under the receiver operating characteristic curve (AUROC), F1, and the area under the precision-recall curve (AUPRC) evaluation metrics. Results on all forecasting datasets are in Table 6.

Table 3: Test accuracy on classification datasets (mean ± std. dev. over 5 random seeds) excluding MIMIC experiments. For each univariate foundation model, per column we bold whichever score is the best and underline the second-best score.

		EthanolConcentration	JapaneseVowels	SelfRegulationSCP1	SelfRegulationSCP2	SpokenArabicDigits	UWaveGestureLibrary
MOMENT	Full	0.283 ± 0.022	0.913 ± 0.016	0.800 ± 0.070	0.503 ± 0.004	0.952 ± 0.003	0.679 ± 0.060
	LoRA	0.326 ± 0.011	0.936 ± 0.014	0.766 ± 0.035	**0.541** ± 0.023	0.975 ± 0.002	**0.853** ± 0.017
	Linear Probing	**0.356** ± 0.009	**0.961** ± 0.003	**0.861** ± 0.002	0.530 ± 0.008	**0.978** ± 0.002	0.761 ± 0.004
	Prompt Tuning	0.327 ± 0.011	0.937 ± 0.012	0.751 ± 0.025	0.538 ± 0.012	0.969 ± 0.005	0.813 ± 0.041
	Gen-P-Tuning	0.323 ± 0.006	0.918 ± 0.020	0.850 ± 0.058	0.526 ± 0.016	0.969 ± 0.005	0.784 ± 0.025
GPT4TS	Full	0.288 ± 0.018	0.918 ± 0.016	0.832 ± 0.021	0.513 ± 0.008	**0.937** ± 0.007	0.729 ± 0.037
	LoRA	0.303 ± 0.017	**0.950** ± 0.014	0.763 ± 0.051	0.531 ± 0.022	0.908 ± 0.022	**0.807** ± 0.035
	Linear Probing	0.295 ± 0.020	0.928 ± 0.007	0.820 ± 0.018	**0.546** ± 0.023	0.918 ± 0.016	0.785 ± 0.019
	Prompt Tuning	**0.312** ± 0.026	0.308 ± 0.075	0.847 ± 0.046	0.517 ± 0.031	0.671 ± 0.043	0.495 ± 0.087
	Gen-P-Tuning	0.297 ± 0.027	0.202 ± 0.110	**0.852** ± 0.017	0.529 ± 0.014	0.646 ± 0.055	0.478 ± 0.091

Table 4: MIMIC Mortality classification test set scores (mean ± std. dev. over 5 random seeds). For each univariate foundation model, per column we bold whichever score is highest and underline the second-best score.

Model	Fine-Tuning Method	Raw Accuracy	AUROC	F1	AUPRC
MOMENT	Full	**0.891** ± 0.012	0.687 ± 0.020	0.508 ± 0.044	0.255 ± 0.038
	LoRA	0.875 ± 0.021	0.720 ± 0.019	0.573 ± 0.050	0.272 ± 0.025
	Linear Probing	0.878 ± 0.013	0.730 ± 0.035	0.544 ± 0.043	0.260 ± 0.018
	Prompt Tuning	0.883 ± 0.012	0.724 ± 0.035	0.576 ± 0.058	0.274 ± 0.020
	Gen-P-Tuning	0.881 ± 0.005	**0.754** ± 0.021	**0.591** ± 0.031	**0.292** ± 0.026
GPT4TS	Full	0.886 ± 0.019	**0.743** ± 0.018	0.524 ± 0.052	**0.309** ± 0.023
	LoRA	0.871 ± 0.017	0.708 ± 0.056	**0.588** ± 0.028	0.254 ± 0.024
	Linear Probing	0.859 ± 0.015	0.737 ± 0.033	0.584 ± 0.037	0.265 ± 0.037
	Prompt Tuning	**0.891** ± 0.013	0.689 ± 0.062	0.471 ± 0.004	0.236 ± 0.022
	Gen-P-Tuning	0.887 ± 0.016	0.708 ± 0.025	0.499 ± 0.033	0.255 ± 0.038

Table 5: MIMIC Phenotyping classification test set scores (mean ± std. dev. over 5 random seeds). For each univariate foundation model, per column we bold whichever score is the best and underline the second-best score.

Model	Fine-Tuning Method	Raw Accuracy	AUROC	F1	AUPRC
MOMENT	Full	0.832 ± 0.007	0.643 ± 0.019	0.070 ± 0.024	0.276 ± 0.021
	LoRA	0.832 ± 0.007	0.640 ± 0.025	0.085 ± 0.023	0.273 ± 0.027
	Linear Probing	0.830 ± 0.006	0.631 ± 0.026	0.071 ± 0.031	0.264 ± 0.022
	Prompt Tuning	0.832 ± 0.008	0.634 ± 0.012	0.069 ± 0.036	0.268 ± 0.015
	Gen-P-Tuning	**0.835** ± 0.004	**0.666** ± 0.015	**0.135** ± 0.017	**0.294** ± 0.012
GPT4TS	Full	0.823 ± 0.009	0.593 ± 0.014	0.060 ± 0.028	0.234 ± 0.014
	LoRA	0.801 ± 0.015	0.596 ± 0.023	0.107 ± 0.024	**0.241** ± 0.015
	Linear Probing	0.789 ± 0.009	0.555 ± 0.016	**0.129** ± 0.029	0.213 ± 0.015
	Prompt Tuning	0.831 ± 0.010	0.581 ± 0.012	0.024 ± 0.017	0.227 ± 0.014
	Gen-P-Tuning	**0.832** ± 0.003	**0.599** ± 0.010	0.020 ± 0.009	0.231 ± 0.010

Table 6: Test MSE and MAE on forecasting datasets (mean ± std. dev. over 5 random seeds). For each univariate foundation model, per column we bold whichever score is the best and underline the second-best score.

		ETTh1		ETTh2		Exchange		ILI	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MOMENT	Full	0.443 ± 0.006	0.451 ± 0.007	0.378 ± 0.019	0.403 ± 0.009	0.823 ± 0.137	0.677 ± 0.057	3.199 ± 0.102	1.262 ± 0.022
	LoRA	**0.427** ± 0.016	**0.438** ± 0.012	**0.351** ± 0.009	**0.390** ± 0.004	0.595 ± 0.116	0.519 ± 0.038	3.109 ± 0.021	1.176 ± 0.006
	Linear Probing	6.566 ± 0.146	1.702 ± 0.019	2.388 ± 0.117	1.078 ± 0.027	1.431 ± 0.033	0.891 ± 0.013	**2.622** ± 0.036	**1.159** ± 0.011
	Prompt Tuning	0.430 ± 0.011	0.441 ± 0.008	0.400 ± 0.019	0.417 ± 0.011	**0.535** ± 0.099	**0.504** ± 0.052	2.918 ± 0.047	1.200 ± 0.009
	Gen-P-Tuning	0.439 ± 0.009	0.445 ± 0.007	0.369 ± 0.028	0.402 ± 0.019	0.582 ± 0.134	0.542 ± 0.061	3.083 ± 0.080	1.200 ± 0.016
GPT4TS	Full	0.514 ± 0.024	0.507 ± 0.016	0.454 ± 0.020	0.457 ± 0.007	1.865 ± 0.191	1.023 ± 0.056	3.219 ± 0.093	1.240 ± 0.012
	LoRA	0.486 ± 0.021	0.490 ± 0.016	0.453 ± 0.011	0.463 ± 0.004	1.194 ± 0.131	0.808 ± 0.054	3.247 ± 0.337	1.230 ± 0.089
	Linear Probing	0.510 ± 0.023	0.494 ± 0.016	0.438 ± 0.056	0.449 ± 0.026	0.879 ± 0.128	0.678 ± 0.036	3.202 ± 0.303	1.232 ± 0.072
	Prompt Tuning	**0.405** ± 0.007	**0.429** ± 0.006	0.308 ± 0.008	0.357 ± 0.005	**0.716** ± 0.040	0.632 ± 0.029	3.105 ± 0.429	1.253 ± 0.098
	Gen-P-Tuning	0.414 ± 0.005	0.436 ± 0.005	**0.298** ± 0.007	**0.353** ± 0.005	0.752 ± 0.217	**0.615** ± 0.092	**2.939** ± 0.378	**1.209** ± 0.092