# UniTime: A Language-Empowered Unified Model for Cross-Domain Time Series Forecasting

Xu Liu
National University of Singapore
liuxu@comp.nus.edu.sg

Junfeng Hu
National University of Singapore
junfengh@comp.nus.edu.sg

Yuan Li
National University of Singapore
li.yuan@u.nus.edu

Shizhe Diao
The Hong Kong University of Science and Technology
sdiaoaa@ust.hk

Yuxuan Liang*
The Hong Kong University of Science and Technology (Guangzhou)
yuxliang@outlook.com

Bryan Hooi
Roger Zimmermann
National University of Singapore
{bhooi,rogerz}@comp.nus.edu.sg

## ABSTRACT

Multivariate time series forecasting plays a pivotal role in contemporary web technologies. In contrast to conventional methods that involve creating dedicated models for specific time series application domains, this research advocates for a unified model paradigm that transcends domain boundaries. However, learning an effective cross-domain model presents the following challenges. First, various domains exhibit disparities in data characteristics, e.g., the number of variables, posing hurdles for existing models that impose inflexible constraints on these factors. Second, the model may encounter difficulties in distinguishing data from various domains, leading to suboptimal performance in our assessments. Third, the diverse convergence rates of time series domains can also result in compromised empirical performance. To address these issues, we propose UniTime for effective cross-domain time series learning. Concretely, UniTime can flexibly adapt to data with varying characteristics. It also uses domain instructions and a Language-TS Transformer to offer identification information and align two modalities. In addition, UniTime employs masking to alleviate domain convergence speed imbalance issues. Our extensive experiments demonstrate the effectiveness of UniTime in advancing state-of-the-art forecasting performance and zero-shot transferability.

## CCS CONCEPTS

• **Mathematics of computing → Time series analysis**.

## KEYWORDS

Time Series Forecasting, Language Models

*Corresponding author. The code is available at: https://github.com/liuxu77/UniTime.

## 1 INTRODUCTION

The World Wide Web, as a dynamic and ever-evolving ecosystem, relies heavily on the ability to anticipate and adapt to changing patterns and user behaviors. Multivariate time series forecasting, with its capacity to analyze historical data and predict future trends, emerges as a crucial tool in modern web technologies [10, 12, 13, 17, 39]. The capability of accurate forecasts has the potential not only to enhance user experiences but also to drive the development of intelligent web services, such as content recommendations [34], web economics modeling [39], microservice logs analysis [14], as well as early warning systems against emerging threats [13].

Recently, Transformers [32] have achieved exceptional performance in various tasks of natural language processing [18, 29, 30] and computer vision [3, 6, 25], which also triggered significant interest in the time series community [35]. Benefiting from the self-attention mechanism to capture long-range temporal dependencies in sequential data, a multitude of Transformer-based models have been proposed for time series forecasting [4, 20, 23, 24, 28, 36, 38, 42, 44, 45]. This rapid progress has consistently pushed the boundaries of state-of-the-art performance in forecasting benchmarks from diverse application domains, including energy, economics, weather, transportation, and disease predictions.

While these models have shown impressive performance, they employ a strategy of training a dedicated model for each domain (or dataset). We argue that this approach may be overly restrictive and overlooks the potential benefits of training a unified model capable of generalizing across various domains. Such a unified model paradigm has achieved remarkable success in computer vision [19, 27], natural language processing [1, 29], and holds promise in the context of time series modeling. An illustration of the two paradigms are presented in Figure 1.

The advantage of training a cross-domain time series model lies in its ability to leverage abundant data from diverse domains with varying temporal characteristics. This enables the model to learn the underlying commonalities present in time series data, which are intrinsic and shared across domains. For instance, while the specific patterns of seasonality (e.g., daily or weekly) may differ between domains, the fundamental concept of recurring patterns within the data is a shared characteristic. Additionally, the presence of trends (e.g., upward or downward) may vary from one domain to another, but the shared property is the recognition of data evolving over time. By equipping the model with this generalization capability, it stands to benefit from enhanced predictive performance and the

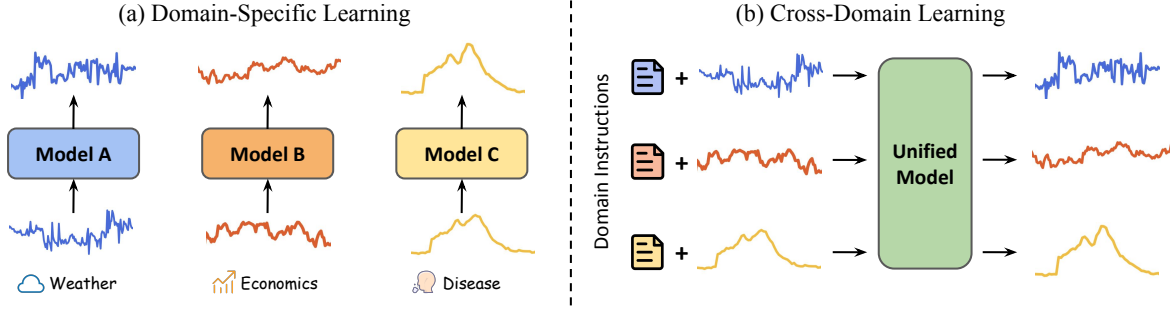(a) Domain-Specific Learning    (b) Cross-Domain Learning



**Figure 1: (a) Specialized models are separately trained on time series domains with notable distribution differences. For instance, weather time series constantly fluctuate due to the chaotic influence of natural factors, while economic data, such as exchange rates, tends to remain relatively stable. Disease data, like seasonal cold patterns, typically demonstrate periodicity over extended time periods. (b) The proposed cross-domain learning approach handles time series data from distinct domains and utilizes natural language as domain instructions to provide domain-specific information.**

ability to transfer its knowledge to previously unseen domains. This potential for broader applicability, improved performance, and streamlined deployment underscores the value of cross-domain time series modeling. However, to effectively learn a unified model for data from diverse domains is technically non-trivial, with the following three challenges.

- **Varying Data Characteristics.** Data from various domains exhibit differences in the number of variables (channels), lengths of histories, and lengths of future predictions. However, existing model designs typically impose rigid constraints on these factors, limiting their ability to generalize across domains. For instance, many approaches employ the channel mixing design [23, 36, 37], which locks the number of input channels to a constant value, making it nearly infeasible to implement a shared encoder capable of handling time series from domains with distinct semantics.
- **Domain Confusion Issue.** When training a model across multiple time series domains, especially when these domains display notable variations in temporal patterns or distributions [24, 38], the model may struggle with discerning and adapting to these differences. This challenge, termed *domain confusion* in this study, results in subpar empirical performance.
- **Domain Convergence Speed Imbalance.** Various time series domains exhibit diverse convergence rates attributed to their unique characteristics. For instance, domains with simple and regular patterns may rapidly reach convergence during model training, and then exhibit a tendency for overfitting, whereas others may require more iterations to achieve convergence. Experimentally, this disparity in learning dynamics leads to a compromise in cross-domain forecasting performance.

To address the aforementioned challenges, this paper introduces **UniTime**, an innovative solution for effectively learning from cross-domain time series data. First, UniTime offers flexibility in its overall design, accommodating time series data with varying characteristics, e.g., input and output lengths. Second, inspired by the recent progress in language instruction-based model tuning [5, 33, 41, 43], we propose the use of human-crafted instructions to furnish the model with explicit domain identification information, alleviating the issue of domain confusion. We further introduce a Language-TS Transformer designed to process both instructions and time series. Thus, time series from different input spaces are aligned to the

common latent space of language models, facilitating cross-domain generalization. Third, we employ masking to mitigate the problem of domain convergence speed imbalance, by constraining the model from acquiring trivial solutions, such as memorizing exclusive data patterns, on domains susceptible to overfitting. Our contributions are summarized below.

- To the best of our knowledge, we present the first attempt to explore the potential of using a unified model for generalization across time series application domains.
- We propose UniTime as a versatile model, which is capable of handling time series data with varying characteristics, distinguishing between different domains, and balancing data with diverse convergence rates.
- Our extensive experiments affirm the effectiveness of UniTime. It attains new state-of-the-art performance on popular time series forecasting benchmarks, and showcases admirable transferability to unseen domains.

## 2 RELATED WORK

**Deep Models for Time Series Forecasting.** Deep learning models with elaborately crafted architectures have demonstrated great promise in time series forecasting. Among them, Transformer-based models have gained widespread recognition due to their exceptional prowess in sequence modeling [35]. However, the self-attention mechanisms in Transformers are known to introduce high computational and memory complexities. Consequently, a plethora of approaches, such as LogTrans [22], Reformer [20], Informer [44], Pyraformer [23] have been proposed to reduce the cost for better efficiency. Another line of research concentrates on capturing the intricate temporal patterns within time series data by leveraging techniques such as seasonal-trend decomposition (Autoformer [38], ETSformer [36], FEDformer [45]) and non-stationary information compensation (NSformer [24]), so as to boost performance. Recently, the community has initiated efforts to develop more versatile methods. For example, TimesNet [37] proposes a generic framework to tackle multiple time series tasks. Following TimesNet, GPT4TS [46] proposes to leverage pretrained language models to process time series signals. However, the above methods still employ separate models for each domain/dataset, limiting their potential to become the foundational architecture for general time series modeling.

**Language Model Powered Cross-Modality Learning.** Recently, there has been a notable surge of interest in the utilization of pre-trained language models to other research fields with distinct modalities, including recommendation systems [11, 41], graph learning [9, 43], and time series modeling [46]. For instance, InstructRec [41] reformulates recommendation tasks into text form, utilizing instructions to enable language models to generate recommendations. GIMLET [43] employs natural language to describe tasks, which not only allows the incorporation of textual knowledge, but also empowers models to accomplish molecule-related tasks using specific instructions. GPT4TS [46] is a relevant work to this study, as it also employs language models to forecast the future. While GPT4TS demonstrates the feasibility of processing time series with language models, it primarily relies on a single modality, namely the time series data itself. It falls short of fully exploiting the powerful language processing capabilities that language models offer, which are pivotal in facilitating cross-domain time series learning.

## 3 PRELIMINARIES

**Problem Definition.** The primary emphasis of this study lies in the development of cross-domain time series models. To this end, we define an observation of a multivariate time series from domain $\tau$ at time step $t$ as $\boldsymbol{x}_\tau^t = \{x_{\tau,1}^t, ..., x_{\tau,c_\tau}^t\} \in \mathbb{R}^{c_\tau}$, where $c_\tau$ represents the number of channels or variables within domain $\tau$. In the context of cross-domain time series forecasting, both the historical and future prediction lengths can vary across domains. Thus, we use $L_\tau$ to denote the lookback window and $T_\tau$ to denote the future prediction range in domain $\tau$, and represent the input and output of the model as $X_\tau^{L_\tau} = \{\boldsymbol{x}_\tau^1, ..., \boldsymbol{x}_\tau^{L_\tau}\} \in \mathbb{R}^{L_\tau \times c_\tau}$ and $\hat{X}_\tau^{T_\tau} = \{\hat{\boldsymbol{x}}_\tau^{L_\tau+1}, ..., \hat{\boldsymbol{x}}_\tau^{L_\tau+T_\tau}\} \in \mathbb{R}^{T_\tau \times c_\tau}$.

**Channel-Mixing v.s. Channel-Independence.** Many time series Transformer models typically adopt a channel-mixing configuration [37, 38, 44, 45]. In this setup, an embedding layer is utilized to process data from all time series channels and project them into a hidden space for multi-channel information fusion. However, this setting poses challenges when attempting to train models across time series domains due to two key issues: (1) the number of channels typically varies among different time series domains, and (2) employing a shared embedding layer to process time series channels from different domains with significantly distinct semantics is impractical. To tackle the problems, our study embraces the channel-independence configuration (recently introduced in PatchTST [28]), which processes each channel individually and provides greater flexibility in handling cross-domain time series.

## 4 THE UNITIME MODEL

In this section, we present the proposed UniTime model, an innovative and generic solution designed for end-to-end learning with cross-domain time series data. Figure 2 provides an overview of the UniTime model, which comprises three primary components: a time series tokenizer to preprocess time series raw signals and prepare the time series tokens, a Language-TS Transformer for domain identification and the alignment of two modalities (text and time series), and a decoder for prediction generation. Given our adoption of the channel-independence setting, we next offer a detailed description of each model component from the perspective of a univariate time series from an arbitrary application domain. Formally, we denote the $i$-th univariate time series from domain $\tau$ with length $L_\tau$ as $\boldsymbol{x}_{\tau,i}^{L_\tau} = \{x_{\tau,i}^1, ..., x_{\tau,i}^{L_\tau}\} \in \mathbb{R}^{L_\tau}$.

### 4.1 Time Series Tokenizer

We propose a time series tokenizer to generate the time series tokens from raw series signals. These tokens will be fed into the proposed Language-TS Transformer, described in next section. Our time series tokenizer involves two sub-modules.

**Time Series Patching.** Recognizing that individual time points lack sufficient semantic meaning like a word in a sentence, we employ patching techniques, as seen in ViT [6] and PatchTST [28], to aggregate adjacent time series into tokens. This helps capture local semantic information in time series, and also reduces the computational overhead when processing long input sequences.

Before patching, we preprocess the raw time series through three steps: (1) masking by a binary vector containing zeros and ones (explained later), (2) series stationarization to mitigate distribution shifts [24, 37], and (3) series padding, which involves duplicating the last value of the original sequence to ensure proper patching. We then segment each univariate time series $\boldsymbol{x}_{\tau,i}^{L_\tau}$ into tokens, which may or may not overlap each other, depending on the specific choice. Concretely, let $P$ denote the time series token length and $S_\tau$ represent the stride value (the non-overlapping distance between the starting point of two consecutive tokens). The patching process generates a sequence of tokens $X_{\tau,i}^{N_\tau} \in \mathbb{R}^{N_\tau \times P}$, where $N_\tau$ is the resulting number of tokens, and $N_\tau = \lceil \frac{L_\tau - P}{S_\tau} \rceil + 1$.

We then employ a shared and learnable linear projection to embed the tokens of each domain to a hidden space $Z_{\tau,i}^{N_\tau} \in \mathbb{R}^{N_\tau \times D}$, where $D$ is set to match that of the Transformer used later. It is worth mentioning that the token size $P$ is fixed and shared across domains due to the usage of the linear projection. The stride value $S_\tau$, on the other hand, is adaptable and depends on the historical observation lengths in each domain.

**Masking & Gated Fusion.** Different time series domains manifest varying convergence rates due to their inherent characteristics. For example, domains with simple and regular patterns may converge swiftly, followed by a tendency to overfit, while others may demand more iterations to achieve convergence. Such an imbalanced learning process results in compromised cross-domain forecasting performance. To alleviate this problem, we propose to employ masking to compel the model to depend only on partial input. Consequently, the model is constrained from learning trivial solutions (e.g., simply memorizing the exclusive patterns of data) on domains that are prone to overfitting, promoting the acquisition of more robust and generalizable representations.

Concretely, for each time series channel, we first generate a binary mask vector $\boldsymbol{m}_{\tau,i}^{L_\tau} \in \{0, 1\}^{L_\tau}$, where the value 0 indicates the specific time steps to be masked, and the ratio of zeros is specified by a parameter $r_m$. This mask vector has two usages: (1) masking the raw time series signals $\boldsymbol{x}_{\tau,i}^{L_\tau}$, and (2) serving as a binary indicator to make the model aware of which positions are masked. To achieve the second usage, the mask vector needs to undergo a process similar to that of the time series signals, i.e., padding and patching.
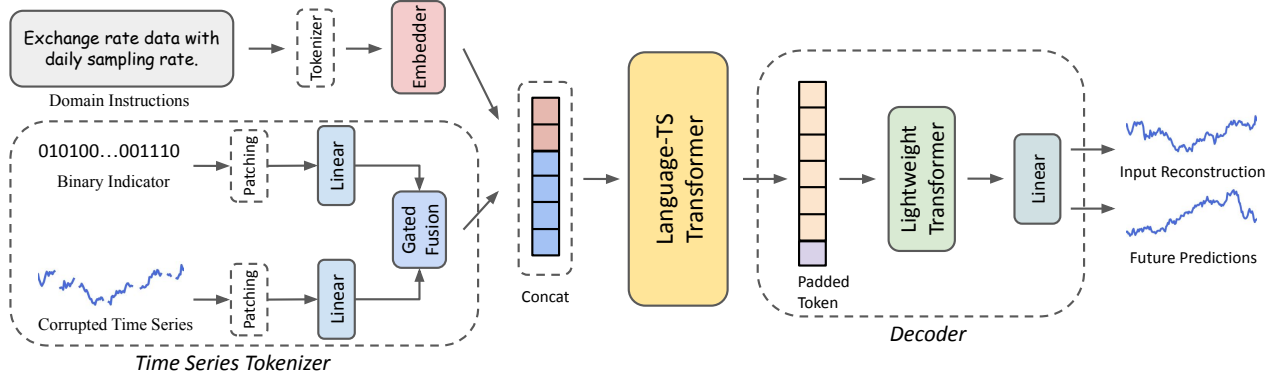
**Figure 2: UniTime overview from the perspective of a univariate time series.**

Subsequently, we apply a linear projection to map it into the hidden space, denoted by $\boldsymbol{M}_{\tau,i}^{N_\tau} \in \mathbb{R}^{N_\tau \times D}$. Then we perform a gated fusion operation to integrate its information with the time series tokens, in order to enhance the model's awareness of which specific information can be used to generate the predictions. Formally,

$$Z_{\tau,i}^{N_\tau} = Gate \odot Z_{\tau,i}^{N_\tau} + (1 - Gate) \odot M_{\tau,i}^{N_\tau} \tag{1}$$

$$Gate = \sigma(Z_{\tau,i}^{N_\tau} W_{g1} + M_{\tau,i}^{N_\tau} W_{g2} + b_g) \tag{2}$$

where $\boldsymbol{W}_{g1}, \boldsymbol{W}_{g2}, \boldsymbol{b}_g$ are learnable parameters and $\sigma(\cdot)$ is a sigmoid function.

## 4.2 Language-TS Transformer

**Motivation.** When training a model across time series domains, especially when these domains exhibit significant differences in temporal patterns or distributions [24, 38], the model may encounter challenges in distinguishing and generalizing between them. This issue, which we refer to as *domain confusion*, leads to poor forecasting performance in our empirical evaluations. In this study, we propose the use of domain instructions to offer explicit domain identification information to the model, facilitating the model to discern the source of each time series and adapt its forecasting strategy accordingly. The domain instructions are essentially sentences describing each domain's data. They are also crafted by humans to incorporate human prior knowledge of the data. Moreover, we propose the use of a Language-TS Transformer to learn joint representations from domain instructions and time series, which enables cross-domain generalization by aligning the time series from various input spaces to the common latent space of the language models.

**Model Design.** In this study, we leverage a pretrained language model to unify language and time series modalities. It is important to note that various language models with different architectures are available, including BERT [18], T5 [30], and GPT2 [29]. Given the autoregressive nature of time series data, we opt for GPT2 as our backbone model, which employs causal masking to preserve the temporal order of inputs. Moreover, it is crucial to consider the order of language and time series when using causal masking. If we place the time series data first, the Transformer won't have access to the domain instructions while processing the time series. This weakens the utility of the text information. Therefore, we choose to position the instructions before the time series data, enabling

the model to directly leverage contextual identifiers to enhance its cross-domain forecasting performance.

Formally, let $e_\tau$ denote the instruction from domain $\tau$ with length $I_\tau$ and $\boldsymbol{E}_{\tau,i}^{I_\tau} \in \mathbb{R}^{I_\tau \times D}$ denote its embeddings. The input to the proposed Language-TS Transformer is: $\boldsymbol{H}_{\tau,i}^{I_\tau+N_\tau} = \left(\boldsymbol{E}_{\tau,i}^{I_\tau} || Z_{\tau,i}^{N_\tau}\right) + \boldsymbol{W}_{pos}$, where $||$ represents the concatenation operation, and $\boldsymbol{W}_{pos}$ is the learnable positional embeddings from the pretrained language model. Kindly note that the first dimension of $\boldsymbol{H}_{\tau,i}^{I_\tau+N_\tau} \in \mathbb{R}^{(I_\tau+N_\tau)\times D}$ varies across domains. This variability is feasible due to Transformer's capability to handle inputs of different lengths. Then we feed $\boldsymbol{H}_{\tau,i}^{I_\tau+N_\tau}$ into $L_{lm}$ Transformer layers with causal attention, whose weights are initialized from GPT2 [29]. We change the superscript of $\boldsymbol{H}_{\tau,i}^{I_\tau+N_\tau}$ to denote the layer index temporarily, and for layer $l = 1, ..., L_{lm}$, the forward process is:

$$\tilde{H}_{\tau,i}^{l-1} = \text{LN}(\text{MSA}(H_{\tau,i}^{l-1})) + H_{\tau,i}^{l-1} \tag{3}$$

$$H_{\tau,i}^l = \text{LN}(\text{MLP}(\tilde{H}_{\tau,i}^{l-1})) + \tilde{H}_{\tau,i}^{l-1} \tag{4}$$

where LN, MSA, and MLP denote a layer normalization, a multi-head self-attention, and a multi-layer perceptron, respectively. Within the MSA, the causal attention is formalized as:

$$\text{Attention}(H_{\tau,i}^{l-1}) = \text{softmax}\left(\frac{Q^l(K^l)^T}{\sqrt{d_k}} + C\right)V^l \tag{5}$$

$$C = \begin{cases} 0, & \text{if position } i \text{ is before } j \\ -\infty, & \text{otherwise} \end{cases} \tag{6}$$

where $Q^l, K^l, V^l$ are the query, key, and value matrices at layer $l$ derived from $H_{\tau,i}^{l-1}$, $d_k$ is the dimension of key, and C is a causal mask matrix.

## 4.3 Decoder

Employing a linear layer to directly produce long-term forecasting results has demonstrated great promise [28, 37, 40], outperforming the traditional iterative approach that is susceptible to substantial error accumulation effects. However, recall that the output of the Language-TS Transformer $\boldsymbol{H}_{\tau,i}^{I_\tau+N_\tau} \in \mathbb{R}^{(I_\tau+N_\tau)\times D}$, which serves as the input to the linear layer, exhibits variations in token lengths. Moreover, the predictive lengths can also vary significantly across diverse domains. These two sources of variability pose a challenge, making it impractical to apply the linear layer directly.

To address this problem, we introduce a maximum token length parameter $R$ and initialize a learnable padding token to ensure consistent sequence lengths across domains. Specifically, we append the padding token repeatedly to $H_{\tau,i}^{I_\tau+N_\tau}$ until the sequence reaches the length of $R$. Then we employ a lightweight Transformer with $L_{light}$ ($L_{light} \ll L_{lm}$) layers to process the padding result. This step serves to inform the other tokens about the presence of the padding token. Finally, we flatten the lightweight Transformer output $\bar{H}_{\tau,i}^R \in \mathbb{R}^{R \times D}$ and utilize a linear layer with a maximum predictive length parameter $O$ to generate predictions. The entire procedure is formalized as follows:

$$\bar{H}_{\tau,i}^R = \text{LightTrans}(\text{Pad}(H_{\tau,i}^{I_\tau+N_\tau})) \tag{7}$$

$$\hat{x}_{\tau,i}^O = \text{Linear}(\text{Flatten}(\bar{H}_{\tau,i}^R)) \tag{8}$$

Note that our model will always generate $O$ values during forecasting. For domains whose predictive length $T_\tau$ is less than $O$, we truncate the first $T_\tau$ values in $\hat{x}_{\tau,i}^O$ as the forecasting outcomes.

## 4.4 Model Training

**Training Objective.** We utilize the widely used mean squared error to assess the disparity between the prediction and the ground truth. Moreover, we simultaneously predict future values and reconstruct past histories, encouraging the model to align its predictions with the observed historical trends [2]. The overall objective loss in domain $\tau$ is averaged over $c_\tau$ channels, and we get:

$$\mathcal{L}_\tau = \frac{1}{c_\tau} \sum_{i=1}^{c_\tau} (\frac{1}{T_\tau} ||\hat{x}_{\tau,i}^{T_\tau} - x_{\tau,i}^{T_\tau}||_2^2 + \frac{1}{L_\tau} ||\hat{x}_{\tau,i}^{L_\tau} - x_{\tau,i}^{L_\tau}||_2^2) \tag{9}$$

**Training Process.** A straightforward approach to cross-domain training involves sequentially feeding each domain's training set to the model during each epoch. However, this method often results in unstable learning and the issue of catastrophic forgetting [7]. To mitigate this problem, we adopt a more granular approach – operating at the batch level. To be specific, we construct batches of data by randomly selecting instances from a pool that encompasses all training data of all involved time series domains. But note that each batch only consists of the data from a single domain. This restriction is due to the varying channel numbers and sequence lengths of each domain. Furthermore, we employ oversampling techniques for domains that have significantly fewer training samples than others. By doing so, we ensure that the model receives ample exposure to these underrepresented domains, preventing them from being overshadowed by the more abundant ones.

## 5 EXPERIMENTS

### 5.1 Experimental Setup

**Datasets.** We extensively assess the proposed UniTime model on eight real-world benchmark datasets, which cover various time series application domains. Here are brief descriptions of the data: (1) **ETT** [44] contains factors used for monitoring electricity transformers between July 2016 and July 2018. ETT involves four subsets: ETTm1, ETTm2, ETTh1 and ETTh2. (2) **Electricity** comprises hourly power consumption of 321 clients from 2012 to 2014. (3) **Exchange** [21] records daily exchange rates of eight different countries ranging from 1990 to 2016. (4) **Weather** is recorded every 10

**Table 1: Summary of datasets.**

| Dataset Name | #Variable | Frequency | #Instances | Application Domain |
|---|---|---|---|---|
| ETTm1/ETTm2 | 7 | 15 mins | 57,507 | Electrical Asset Monitoring |
| ETTh1/ETTh2 | 7 | 1 hour | 14,307 | Electrical Asset Monitoring |
| Electricity | 321 | 1 hour | 26,211 | Electricity Consumption |
| Weather | 21 | 10 mins | 52,603 | Meteorologic Monitoring |
| Exchange | 8 | 1 day | 7,207 | Foreign Exchange Market |
| Illness | 7 | 1 week | 861 | Epidemiological Monitoring |

minutes in the year of 2020. It contains 21 meteorological indicators, such as temperature, humidity, and precipitation. (5) **Illness** includes weekly recorded data on the number of patients with seven influenza-like illnesses between 2002 and 2021. Table 1 provides a summary of the datasets. It can be seen that time series data from various domains exhibit differences in terms of the number of variables, the semantics of those variables, the sampling frequency, and the size of the collected data.

**Baselines.** We include eight state-of-the-art methods for multivariate time series forecasting comparisons, including Informer [44], Autoformer [38], FEDformer [45], NSformer [24], DLinear [40], TimesNet [37], PatchTST [28], and GPT4TS [46], a recent paper that uses language models to process time series data. Note that all these methods train a dedicated model for each evaluated dataset and for each assessed predictive length in their original papers.

**Implementation Details.** We adhere to the same experimental settings as in Wu et al. [37] to ensure a fair comparison: we set the maximum number of epochs to 10 and fix the lookback window length to 36 for the Illness dataset, and 96 for the others. Moreover, we utilize a pretrained GPT2 [29] model as the backbone, with its layer count $L_{lm}$ set at 6, and we do not freeze any of its parameters. For the lightweight Transformer, we configure the $L_{light}$ to 2. The patch length $P$, maximum token length $R$, maximum predictive length $O$, mask ratio $r_m$ are consistently set to 16, 17, 720, and 0.5, respectively. The configuration specifics for each dataset and the results of the hyperparameter studies are provided in Appendix. We train our method via the AdamW optimizer with an initial learning rate of 0.0001. Regarding model selection, we calculate the validation loss for all the datasets involved and then compute an average score. The model that achieves the lowest overall validation loss will be used for testing. Experiments are executed on an NVIDIA A100 80GB GPU.

### 5.2 Main Results

Table 2 presents the overall forecasting performance. We utilize two vertical lines to demarcate the table. The right part of the table signifies that separate models are trained for each dataset and for each specific predictive length. To illustrate, for the ETTm1 dataset, four distinct models are created to predict four different future lengths: 96, 192, 336, and 720. On the left side of the table, models are trained across datasets and consistently generate 720 future values. When evaluating performance for a setting shorter than 720 entries, such as 96, we simply take the first 96 values within the 720-value output. According to the table, the proposed UniTime model demonstrates remarkable improvements over the baseline models that are also trained across datasets, securing the best performance in 79 out of 80 entries. Moreover, UniTime delivers competitive results when compared to models trained individually on each

**Table 2: Forecasting performance comparisons. The input sequence length is set to 36 for the Illness dataset and 96 for the others. The predictive lengths are set to {24, 36, 48, 60} for Illness, and {96, 192, 336, 720} for others. Avg is averaged over all predictive lengths. Note that we bold the best performance among models trained across datasets, which is on the left-hand side of the two vertical lines, and we bold and underline the best performance for the entire row.**

| Method | | UniTime MSE | MAE | GPT4TS† MSE | MAE | PatchTST† MSE | MAE | GPT4TS* MSE | MAE | PatchTST* MSE | MAE | TimesNet MSE | MAE | DLinear MSE | MAE | NSformer MSE | MAE | FEDformer MSE | MAE | Autoformer MSE | MAE | Informer MSE | MAE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Models Trained Across Datasets** | | | | | | **Models Trained on Each Dataset** | | | | | | | | | | | | | | | |
| ETTm1 | 96 | **0.322** | **0.363** | 0.509 | 0.463 | 0.927 | 0.604 | 0.335 | 0.369 | 0.344 | 0.373 | 0.338 | 0.375 | 0.345 | 0.372 | 0.386 | 0.398 | 0.379 | 0.419 | 0.505 | 0.475 | 0.672 | 0.571 |
| | 192 | **0.366** | **0.387** | 0.537 | 0.476 | 0.964 | 0.620 | 0.374 | **0.385** | 0.367 | 0.386 | 0.374 | 0.387 | 0.380 | 0.389 | 0.459 | 0.444 | 0.426 | 0.441 | 0.553 | 0.496 | 0.795 | 0.669 |
| | 336 | **0.398** | **0.407** | 0.564 | 0.488 | 1.041 | 0.656 | 0.407 | **0.406** | **0.392** | 0.407 | 0.410 | 0.411 | 0.413 | 0.413 | 0.495 | 0.464 | 0.445 | 0.459 | 0.621 | 0.537 | 1.212 | 0.871 |
| | 720 | **0.454** | **0.440** | 0.592 | 0.504 | 0.950 | 0.636 | 0.469 | 0.442 | 0.464 | 0.442 | 0.478 | 0.450 | 0.474 | 0.453 | 0.585 | 0.516 | 0.543 | 0.490 | 0.671 | 0.561 | 1.166 | 0.823 |
| | Avg | **0.385** | **0.399** | 0.551 | 0.483 | 0.971 | 0.629 | 0.396 | 0.401 | 0.392 | 0.402 | 0.400 | 0.406 | 0.403 | 0.407 | 0.481 | 0.456 | 0.448 | 0.452 | 0.588 | 0.517 | 0.961 | 0.734 |
| ETTm2 | 96 | **0.183** | **0.266** | 0.229 | 0.304 | 0.240 | 0.318 | 0.190 | 0.275 | **0.177** | **0.260** | 0.187 | 0.267 | 0.193 | 0.292 | 0.192 | 0.274 | 0.203 | 0.287 | 0.255 | 0.339 | 0.365 | 0.453 |
| | 192 | **0.251** | **0.310** | 0.287 | 0.338 | 0.301 | 0.352 | 0.253 | 0.313 | **0.246** | **0.305** | 0.249 | 0.309 | 0.284 | 0.362 | 0.280 | 0.339 | 0.269 | 0.328 | 0.281 | 0.340 | 0.533 | 0.563 |
| | 336 | **0.319** | **0.351** | 0.337 | 0.367 | 0.367 | 0.391 | 0.321 | 0.360 | **0.305** | **0.343** | 0.321 | 0.351 | 0.369 | 0.427 | 0.334 | 0.361 | 0.325 | 0.366 | 0.339 | 0.372 | 1.363 | 0.887 |
| | 720 | **0.420** | **0.410** | 0.430 | 0.416 | 0.451 | 0.432 | 0.411 | 0.406 | 0.410 | 0.405 | **0.408** | **0.403** | 0.554 | 0.522 | 0.417 | 0.413 | 0.421 | 0.415 | 0.433 | 0.432 | 3.379 | 1.338 |
| | Avg | **0.293** | **0.334** | 0.321 | 0.356 | 0.340 | 0.373 | 0.294 | 0.339 | **0.285** | **0.328** | 0.291 | 0.333 | 0.350 | 0.401 | 0.306 | 0.347 | 0.305 | 0.349 | 0.327 | 0.371 | 1.410 | 0.810 |
| ETTh1 | 96 | **0.397** | 0.418 | 0.449 | 0.424 | 0.409 | **0.403** | 0.398 | 0.424 | 0.404 | 0.413 | 0.384 | 0.402 | 0.386 | **0.400** | 0.513 | 0.491 | **0.376** | 0.419 | 0.449 | 0.459 | 0.865 | 0.713 |
| | 192 | **0.434** | **0.439** | 0.503 | 0.453 | 0.467 | 0.444 | 0.449 | **0.427** | 0.454 | 0.440 | 0.436 | 0.429 | 0.437 | 0.432 | 0.534 | 0.504 | **0.420** | 0.448 | 0.500 | 0.482 | 1.008 | 0.792 |
| | 336 | **0.468** | **0.457** | 0.540 | 0.477 | 0.509 | 0.472 | 0.492 | 0.466 | 0.497 | 0.462 | 0.491 | 0.469 | 0.481 | 0.459 | 0.588 | 0.535 | **0.459** | 0.465 | 0.521 | 0.496 | 1.107 | 0.809 |
| | 720 | **0.469** | **0.477** | 0.515 | 0.489 | 0.503 | 0.485 | 0.487 | 0.483 | 0.496 | 0.481 | 0.521 | 0.500 | 0.519 | 0.516 | 0.643 | 0.616 | 0.506 | 0.507 | 0.514 | 0.512 | 1.181 | 0.865 |
| | Avg | **0.442** | **0.448** | 0.502 | 0.461 | 0.472 | 0.451 | 0.457 | 0.450 | 0.463 | 0.449 | 0.458 | 0.450 | 0.456 | 0.452 | 0.570 | 0.537 | **0.440** | 0.460 | 0.496 | 0.487 | 1.040 | 0.795 |
| ETTh2 | 96 | **0.296** | **0.345** | 0.303 | 0.349 | 0.314 | 0.361 | 0.312 | 0.360 | 0.312 | 0.358 | 0.340 | 0.374 | 0.333 | 0.387 | 0.476 | 0.458 | 0.358 | 0.397 | 0.346 | 0.388 | 3.755 | 1.525 |
| | 192 | **0.374** | **0.394** | 0.391 | 0.399 | 0.407 | 0.411 | 0.387 | 0.405 | 0.397 | 0.408 | 0.402 | 0.414 | 0.477 | 0.476 | 0.512 | 0.493 | 0.429 | 0.439 | 0.456 | 0.452 | 5.602 | 1.931 |
| | 336 | **0.415** | **0.427** | 0.422 | 0.428 | 0.437 | 0.443 | 0.424 | 0.437 | 0.435 | 0.440 | 0.452 | 0.452 | 0.594 | 0.541 | 0.552 | 0.551 | 0.496 | 0.487 | 0.482 | 0.486 | 4.721 | 1.835 |
| | 720 | **0.425** | **0.444** | 0.429 | 0.449 | 0.434 | 0.448 | 0.433 | 0.453 | 0.436 | 0.449 | 0.462 | 0.468 | 0.831 | 0.657 | 0.562 | 0.560 | 0.463 | 0.474 | 0.515 | 0.511 | 3.647 | 1.625 |
| | Avg | **0.378** | **0.403** | 0.386 | 0.406 | 0.398 | 0.416 | 0.389 | 0.414 | 0.395 | 0.414 | 0.414 | 0.427 | 0.559 | 0.515 | 0.526 | 0.516 | 0.437 | 0.449 | 0.450 | 0.459 | 4.431 | 1.729 |
| Electricity | 96 | **0.196** | **0.287** | 0.232 | 0.321 | 0.198 | 0.290 | 0.197 | 0.290 | 0.186 | **0.269** | **0.168** | 0.272 | 0.197 | 0.282 | 0.169 | 0.273 | 0.193 | 0.308 | 0.201 | 0.317 | 0.274 | 0.368 |
| | 192 | **0.199** | **0.291** | 0.234 | 0.325 | 0.202 | 0.293 | 0.201 | 0.292 | 0.190 | **0.273** | 0.184 | 0.289 | 0.196 | 0.285 | **0.182** | 0.286 | 0.201 | 0.315 | 0.222 | 0.334 | 0.296 | 0.386 |
| | 336 | **0.214** | **0.305** | 0.249 | 0.338 | 0.223 | 0.318 | 0.217 | 0.309 | 0.206 | **0.290** | **0.198** | 0.300 | 0.209 | 0.301 | 0.200 | 0.304 | 0.214 | 0.329 | 0.231 | 0.338 | 0.300 | 0.394 |
| | 720 | **0.254** | **0.335** | 0.289 | 0.366 | 0.259 | 0.341 | 0.253 | 0.339 | 0.247 | 0.322 | **0.220** | **0.320** | 0.245 | 0.333 | 0.222 | 0.321 | 0.246 | 0.355 | 0.254 | 0.361 | 0.373 | 0.439 |
| | Avg | **0.216** | **0.305** | 0.251 | 0.338 | 0.221 | 0.311 | 0.217 | 0.308 | 0.207 | **0.289** | **0.192** | 0.295 | 0.212 | 0.300 | 0.193 | 0.296 | 0.214 | 0.327 | 0.227 | 0.338 | 0.311 | 0.397 |
| Weather | 96 | **0.171** | **0.214** | 0.212 | 0.251 | 0.213 | 0.260 | 0.203 | 0.244 | 0.177 | 0.218 | 0.172 | 0.220 | 0.196 | 0.255 | 0.173 | 0.223 | 0.217 | 0.296 | 0.266 | 0.336 | 0.300 | 0.384 |
| | 192 | **0.217** | **0.254** | 0.261 | 0.288 | 0.269 | 0.300 | 0.247 | 0.277 | 0.222 | 0.259 | 0.219 | 0.261 | 0.237 | 0.296 | 0.245 | 0.285 | 0.276 | 0.336 | 0.307 | 0.367 | 0.598 | 0.544 |
| | 336 | **0.274** | **0.293** | 0.313 | 0.324 | 0.330 | 0.341 | 0.297 | 0.311 | 0.277 | 0.297 | 0.280 | 0.306 | 0.283 | 0.335 | 0.321 | 0.338 | 0.339 | 0.380 | 0.359 | 0.395 | 0.578 | 0.523 |
| | 720 | **0.351** | **0.343** | 0.386 | 0.372 | 0.404 | 0.389 | 0.368 | 0.356 | 0.352 | 0.347 | 0.365 | 0.359 | **0.345** | 0.381 | 0.414 | 0.410 | 0.403 | 0.428 | 0.419 | 0.428 | 1.059 | 0.741 |
| | Avg | **0.253** | **0.276** | 0.293 | 0.309 | 0.304 | 0.323 | 0.279 | 0.297 | 0.257 | 0.280 | 0.259 | 0.287 | 0.265 | 0.317 | 0.288 | 0.314 | 0.309 | 0.360 | 0.338 | 0.382 | 0.634 | 0.548 |
| Exchange | 96 | **0.086** | **0.209** | 0.142 | 0.261 | 0.137 | 0.260 | 0.091 | 0.212 | 0.109 | 0.236 | 0.107 | 0.234 | 0.088 | 0.218 | 0.111 | 0.237 | 0.148 | 0.278 | 0.197 | 0.323 | 0.847 | 0.752 |
| | 192 | **0.174** | **0.299** | 0.224 | 0.339 | 0.222 | 0.341 | 0.183 | 0.304 | 0.205 | 0.327 | 0.226 | 0.344 | 0.176 | 0.315 | 0.219 | 0.335 | 0.271 | 0.380 | 0.300 | 0.369 | 1.204 | 0.895 |
| | 336 | **0.319** | **0.408** | 0.377 | 0.448 | 0.372 | 0.447 | 0.328 | 0.417 | 0.356 | 0.436 | 0.367 | 0.448 | **0.313** | 0.427 | 0.421 | 0.476 | 0.460 | 0.500 | 0.509 | 0.524 | 1.672 | 1.036 |
| | 720 | **0.875** | **0.701** | 0.939 | 0.736 | 0.912 | 0.727 | 0.880 | 0.704 | 0.888 | 0.716 | 0.964 | 0.746 | **0.839** | **0.695** | 1.092 | 0.769 | 1.195 | 0.841 | 1.447 | 0.941 | 2.478 | 1.310 |
| | Avg | **0.364** | **0.404** | 0.421 | 0.446 | 0.411 | 0.444 | 0.371 | 0.409 | 0.390 | 0.429 | 0.416 | 0.443 | **0.354** | 0.414 | 0.461 | 0.454 | 0.519 | 0.500 | 0.613 | 0.539 | 1.550 | 0.998 |
| Illness | 24 | **2.460** | **0.954** | 3.322 | 1.278 | 4.289 | 1.485 | 2.732 | 1.100 | 2.335 | 0.989 | 2.317 | **0.934** | 2.398 | 1.040 | **2.294** | 0.945 | 3.228 | 1.260 | 3.483 | 1.287 | 5.764 | 1.677 |
| | 36 | **1.998** | **0.912** | 3.696 | 1.374 | 4.360 | 1.510 | 2.664 | 1.063 | 2.561 | 1.035 | 1.972 | 0.920 | 2.646 | 1.088 | **1.825** | **0.848** | 2.679 | 1.080 | 3.103 | 1.148 | 4.755 | 1.467 |
| | 48 | **1.979** | **0.912** | 3.765 | 1.402 | 4.209 | 1.481 | 2.617 | 1.041 | 2.465 | 1.022 | 2.238 | 0.940 | 2.614 | 1.086 | 2.010 | **0.900** | 2.622 | 1.078 | 2.669 | 1.085 | 4.763 | 1.469 |
| | 60 | **2.109** | **0.938** | 3.928 | 1.432 | 3.981 | 1.444 | 2.478 | 1.035 | 2.189 | 0.997 | **2.027** | **0.928** | 2.804 | 1.146 | 2.178 | 0.963 | 2.857 | 1.157 | 2.770 | 1.125 | 5.264 | 1.564 |
| | Avg | **2.137** | **0.929** | 3.678 | 1.372 | 4.210 | 1.480 | 2.623 | 1.060 | 2.388 | 1.011 | 2.139 | 0.931 | 2.616 | 1.090 | **2.077** | **0.914** | 2.847 | 1.144 | 3.006 | 1.161 | 5.137 | 1.544 |
| 1st Count | | 37 | | 0 | | 0 | | 3 | | 13 | | 10 | | 6 | | 7 | | 4 | | 0 | | 0 | |

† means that we modify the baselines' code (e.g., use padding to align input lengths across different domains), and make them train and test in the same way as our method. ∗ indicates that we adopt the official code of the baselines and reset their input sequence length and maximum training epochs number for a fair comparison to other methods. Other results are from TimesNet [37].

dataset, as demonstrated by improving 37 out of 80 entries to the new state-of-the-art. This outcome validates the effectiveness of our model in handling time series data with diverse characteristics, such as sampling frequency and periodicity.

## 5.3 Ablation Studies

We conduct ablation studies on five variants of UniTime and summarize the results in Table 3. Firstly, *w/o instructions* causes a significant drop in performance across all datasets, with the most pronounced effects on ETTm1 and Illness. This emphasizes the critical role of domain instructions in providing identification information to the model. To further investigate the domain confusion issue, we conduct a comparison between the hidden representations of UniTime *w/o instructions* and UniTime *w/ instructions* using the T-SNE visualization tool [31]. Specifically, for each dataset, we randomly select 100 samples from their respective test sets, and visualize the hidden representations produced by the Language-TS Transformer. In Figure 4, we can observe that in the absence of instructions, the

**Table 3: Ablation of method designs. Due to page limit, for each dataset, we report the average value over all predictive lengths.**

| Variant | ETTm1 | | ETTm2 | | ETTh1 | | ETTh2 | | Electricity | | Weather | | Exchange | | Illness | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| UniTime | **0.385** | **0.399** | 0.293 | 0.334 | 0.442 | 0.448 | **0.378** | **0.403** | 0.216 | 0.305 | **0.253** | **0.276** | **0.364** | **0.404** | **2.137** | **0.929** |
| w/o instructions | 0.479 | 0.461 | 0.311 | 0.349 | 0.466 | 0.449 | 0.397 | 0.409 | 0.221 | 0.310 | 0.283 | 0.307 | 0.389 | 0.428 | 2.381 | 1.041 |
| w/o masking | 0.390 | 0.408 | **0.286** | **0.332** | 0.459 | 0.461 | 0.380 | 0.406 | **0.210** | **0.298** | 0.257 | 0.280 | 0.379 | 0.417 | 2.606 | 1.112 |
| w/o LightTrans | 0.392 | 0.402 | 0.295 | 0.336 | 0.443 | **0.445** | 0.382 | 0.405 | 0.222 | 0.308 | 0.261 | 0.284 | 0.375 | 0.414 | 2.303 | 0.998 |
| w/o reconstruction | 0.392 | 0.405 | 0.294 | 0.336 | **0.439** | 0.447 | 0.383 | 0.407 | 0.220 | 0.312 | 0.259 | 0.281 | 0.383 | 0.417 | 2.197 | 0.956 |
| w/o all | 0.487 | 0.462 | 0.313 | 0.352 | 0.469 | 0.459 | 0.391 | 0.407 | 0.219 | 0.308 | 0.276 | 0.297 | 0.395 | 0.430 | 2.479 | 1.084 |
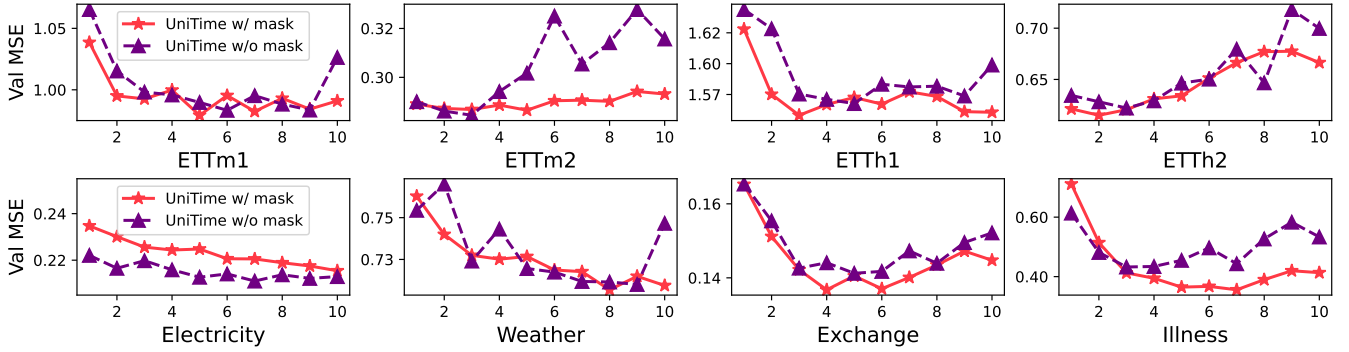


**Figure 3: Visualization of the validation loss during model training. The x-axis denotes the training epoch number.**
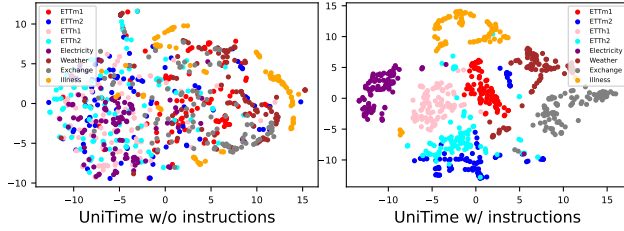


**Figure 4: T-SNE visualization of the hidden representations.**

representations of different domains are mixed together, whereas with the inclusion of instructions, they exhibit clear clustering-like patterns. This observation confirms the existence of domain confusion, and underscores the effectiveness of instructions as a tool to address it. Note that in the visualization of UniTime *w/ instructions*, the clusters of ETTm1, ETTm2, ETTh1, ETTh2 are close to each other. This proximity is attributed to the fact that they belong to the same domain and thus share underlying temporal characteristics.

The results obtained under the *w/o masking* setting reveal that while the model performs satisfactorily on some datasets, the performance on other datasets is significantly degraded, especially on Illness. This decline can be attributed to an imbalanced cross-domain learning process that occurs when masking is disabled. To illustrate this point further, we have plotted the changes in validation loss in Figure 3. Recall that the overall validation loss across all domains is a critical factor during the model selection process. When masking is turned off, the datasets display varying convergence speeds. For example, ETTm2, ETTh2, Exchange, and Illness experience severe overfitting beyond the 4th epoch, while others require more epochs to reach convergence. This lack of balance poses challenges in the model selection process when aiming to choose a model that performs well across all datasets. However,

when masking is enabled, the majority of loss curves do not demonstrate an overfitting trend. Instead, they converge at a later phase and exhibit increased stability. Such a balanced learning environment allows the model to be selected in the later phase of training, leading to superior overall performance.

Furthermore, *w/o LightTrans* and *w/o reconstruction* mean that we remove the light Transformer after the language model and disable the auxiliary reconstruction loss, respectively. The results show that both of them are effective in boosting the overall performance. Finally, the setting of *w/o all* turns off all the aforementioned designs, resulting in degraded performance across all datasets.

## 5.4 Zero-Shot Transferability Analysis

**Setups.** In this part, we delve into the transferability of our methods and baseline models from the source (training) domains to the target (unseen) domains. Specifically, we first train the models on the datasets of ETTh1, ETTm1, and ETTm2. Then we assess their performance in both in-domain transfer and out-domain transfer scenarios through zero-shot testing. This testing is conducted on ETTh2 (hailing from the same domain as the source), Electricity (a different domain with some underlying relations to the source domain), and Weather (representing a completely unrelated domain).

**Transfer Protocol.** Before executing zero-shot transfers with our UniTime model, a preliminary step involves selecting the appropriate domain instructions for the unseen domain. The rationale behind this is that if two domains share common patterns, they may favor similar instructions for their identification. In this study, we propose an instruction selection protocol that hinges on the instructions visible to the models during training. Specifically, we leverage the model input, namely historical observations, and partition them into two parts: the first part is fed into the model to generate the predictions and the second part is utilized to compute the forecasting loss. This loss calculation offers insights into which

**Table 4: Results of design choices related to the language model. We report the average results over all predictive lengths.**

| Variant | ETTm1 | | ETTm2 | | ETTh1 | | ETTh2 | | Electricity | | Weather | | Exchange | | Illness | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| UniTime | **0.385** | **0.399** | **0.293** | **0.334** | 0.442 | 0.448 | **0.378** | **0.403** | **0.216** | **0.305** | **0.253** | **0.276** | **0.364** | **0.404** | **2.137** | **0.929** |
| TS-Text | 0.391 | 0.403 | 0.295 | 0.337 | 0.446 | 0.452 | 0.381 | 0.406 | 0.220 | 0.309 | 0.261 | 0.284 | 0.381 | 0.414 | 2.258 | 1.018 |
| Random Init | 0.404 | 0.411 | 0.297 | 0.339 | 0.446 | 0.451 | 0.379 | 0.404 | 0.220 | 0.309 | 0.260 | 0.281 | 0.374 | 0.413 | 2.336 | 1.043 |
| Freeze PLM | 0.398 | 0.410 | 0.297 | 0.338 | 0.444 | 0.452 | 0.378 | 0.405 | 0.224 | 0.314 | 0.262 | 0.283 | 0.373 | 0.409 | 2.481 | 1.078 |
| FPT PLM | 0.391 | 0.407 | 0.295 | 0.336 | **0.438** | **0.446** | 0.378 | 0.403 | 0.220 | 0.310 | 0.260 | 0.283 | 0.376 | 0.412 | 2.286 | 1.028 |

**Table 5: Zero-shot transferability comparisons.**

| Method | | UniTime | | GPT4TS | | PatchTST | | Repeat | |
|---|---|---|---|---|---|---|---|---|---|
| | | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh2 | 96 | **0.306** | **0.352** | 0.316 | 0.361 | 0.332 | 0.371 | 0.432 | 0.422 |
| | 192 | **0.389** | **0.401** | 0.400 | 0.410 | 0.422 | 0.421 | 0.534 | 0.473 |
| | 336 | **0.424** | **0.434** | 0.430 | 0.439 | 0.462 | 0.455 | 0.597 | 0.511 |
| | 720 | **0.433** | **0.450** | 0.442 | 0.461 | 0.467 | 0.469 | 0.594 | 0.519 |
| | Avg | **0.388** | **0.409** | 0.397 | 0.418 | 0.421 | 0.429 | 0.539 | 0.481 |
| Electricity | 96 | **0.409** | **0.481** | 0.448 | 0.520 | 0.529 | 0.562 | 1.588 | 0.945 |
| | 192 | **0.410** | **0.484** | 0.443 | 0.517 | 0.507 | 0.550 | 1.596 | 0.951 |
| | 336 | **0.439** | **0.504** | 0.462 | 0.526 | 0.536 | 0.566 | 1.618 | 0.961 |
| | 720 | **0.487** | **0.531** | 0.494 | 0.542 | 0.563 | 0.581 | 1.647 | 0.975 |
| | Avg | **0.436** | **0.500** | 0.462 | 0.526 | 0.534 | 0.565 | 1.612 | 0.958 |
| Weather | 96 | **0.210** | 0.262 | 0.223 | 0.271 | 0.235 | 0.277 | 0.259 | **0.254** |
| | 192 | **0.264** | 0.303 | 0.287 | 0.319 | 0.293 | 0.320 | 0.309 | **0.292** |
| | 336 | **0.326** | **0.334** | 0.347 | 0.357 | 0.351 | 0.356 | 0.376 | 0.338 |
| | 720 | **0.402** | **0.382** | 0.432 | 0.409 | 0.427 | 0.404 | 0.465 | 0.394 |
| | Avg | **0.301** | **0.320** | 0.322 | 0.339 | 0.327 | 0.339 | 0.352 | **0.320** |

instruction is most suitable for the unseen data. Experimentally, we conduct this protocol on 0.5% of test samples to determine the instructions to be used. We then apply the selected instruction to all the test samples.

**Results.** Table 5 displays the results of zero-shot testing, with the last column labeled "Repeat" serving as a baseline that simply utilizes the last value of histories as the forecast value for all future time steps. The table clearly illustrates that UniTime consistently outperforms the baselines across the majority of cases, affirming the effectiveness of incorporating instructions. Furthermore, in accordance with our instruction selection protocol, all three zero-shot datasets opt for instructions derived from the data of ETTh1. This choice is well-founded, particularly for the ETTh2 dataset, as it exhibits strong connections with ETTh1. The reason for the Electricity and Weather datasets opting for ETTh1's instruction likely stems from their similar underlying patterns, which lends further support to our approach's adaptability across diverse domains.

### 5.5 Exploration Studies on Language Models

This section conducts further investigations into the factors associated with the language model.

**Input Order.** We explore the effects of altering the input order by placing the time series data before the instructions. In this configuration, time series tokens are unable to attend to the instruction tokens due to the presence of a causal mask. As shown in the second row of Table 4, it's clear that UniTime outperforms this variant with the changed order. The relatively small performance gap is due to our use of a decoder following the Language-TS Transformer. This decoder uses information from the instruction tokens to generate predictions, mitigating the impact of the altered input order.

**Initialization.** In this setting, we forego the use of pretrained weights from GPT-2, opting instead for randomly initialized weights. As evident from the third row of Table 4, we can see that the performance of this configuration on all datasets is inferior to that of our default model. This observation indicates the superiority of pretrained weights, which have been learned from a vast language corpus, in effectively processing textual information.

**Tunability.** In our main results, we fully tuned the pretrained language model (PLM). In this part, we explore alternative approaches: freezing the entire language model, referred to as "Freeze PLM", and freezing the majority of parameters in the language model, denoted as "FPT PLM" [26, 46]. To be specific, the FPT method tunes only the positional embeddings and layer normalization components of the model while keeping the other components, such as self-attention and feed-forward networks, frozen.

The experimental results are summarized in the last two rows of Table 4. Firstly, it is evident that fully tuning the model yields the best performance, followed by the cases of FPT and Freeze. Secondly, a noteworthy finding is that the performance remains relatively strong even when we freeze the entire language model. This outcome suggests that the language model possesses the capability to process time series tokens and generate reasonable hidden representations. This interesting phenomenon is also observed by a recent study [46], and they attribute such universal computing ability to the self-attention modules of a trained Transformer, which behaves similarly to principal component analysis. Thirdly, considering that only a minor subset of parameters requires tuning under the FPT method, it strikes a good balance between performance and efficiency. This makes it an attractive choice when computational resources are limited.

## 6 CONCLUSION

This paper delves into an innovative and pivotal learning paradigm: developing a unified forecasting model capable of accommodating diverse time series application domains. We identify the challenges in constructing such a unified model and propose the novel UniTime to address them accordingly. Our extensive evaluations confirm the effectiveness of UniTime in advancing state-of-the-art forecasting performance and zero-shot transferability. We believe that this work represents a significant step towards building a foundation model for general time series forecasting.

# REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in neural information processing systems*. 1877–1901.

[2] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. In *Advances in Neural Information Processing Systems*. 17766–17778.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European conference on computer vision*. 213–229.

[4] Weiqi Chen, Wenwei Wang, Bingqing Peng, Qingsong Wen, Tian Zhou, and Liang Sun. 2022. Learning to rotate: Quaternion transformer for complicated periodical time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 146–156.

[5] Shizhe Diao, Rui Pan, Hanze Dong, Ka Shun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. 2023. Lmflow: An extensible toolkit for finetuning and inference of large foundation models. *arXiv preprint arXiv:2306.12420* (2023).

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[7] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211* (2013).

[8] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2023. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems*.

[9] Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. 2023. Explanations as Features: LLM-Based Features for Text-Attributed Graphs. *arXiv preprint arXiv:2305.19523* (2023).

[10] Min Hou, Chang Xu, Zhi Li, Yang Liu, Weiqing Liu, Enhong Chen, and Jiang Bian. 2022. Multi-Granularity Residual Learning with Confidence Estimation for Time Series Prediction. In *Proceedings of the ACM Web Conference 2022*. 112–121.

[11] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems. *arXiv preprint arXiv:2305.08845* (2023).

[12] Sheo Yon Jhin, Jaehoon Lee, Minju Jo, Seungji Kook, Jinsung Jeon, Jihyeon Hyeong, Jayoung Kim, and Noseong Park. 2022. Exit: Extrapolation and interpolation-based neural controlled differential equations for time-series classification and forecasting. In *Proceedings of the ACM Web Conference 2022*. 3102–3112.

[13] Renhe Jiang, Zhaonan Wang, Yudong Tao, Chuang Yang, Xuan Song, Ryosuke Shibasaki, Shu-Ching Chen, and Mei-Ling Shyu. 2023. Learning Social Meta-knowledge for Nowcasting Human Mobility in Disaster. In *Proceedings of the ACM Web Conference 2023*. 2655–2665.

[14] Xinrui Jiang, Yicheng Pan, Meng Ma, and Ping Wang. 2023. Look Deep into the Microservice System Anomaly through Very Sparse Logs. In *Proceedings of the ACM Web Conference 2023*. 2970–2978.

[15] Ming Jin, Qingsong Wen, Yuxuan Liang, Chaoli Zhang, Siqiao Xue, Xue Wang, James Zhang, Yi Wang, Haifeng Chen, Xiaoli Li, et al. 2023. Large models for time series and spatio-temporal data: A survey and outlook. *arXiv preprint arXiv:2310.10196* (2023).

[16] Ming Jin, Yifan Zhang, Wei Chen, Kexin Zhang, Yuxuan Liang, Bin Yang, Jindong Wang, Shirui Pan, and Qingsong Wen. 2024. Position Paper: What Can Large Language Models Tell Us about Time Series Analysis. *arXiv preprint arXiv:2402.02713* (2024).

[17] Harshavardhan Kamarthi, Lingkai Kong, Alexander Rodríguez, Chao Zhang, and B Aditya Prakash. 2022. CAMul: Calibrated and Accurate Multi-view Time-Series Forecasting. In *Proceedings of the ACM Web Conference 2022*. 3174–3185.

[18] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*. 2.

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643* (2023).

[20] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

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

[22] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in neural information processing systems*.

[23] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2022. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*.

[24] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting. In *Advances in Neural Information Processing Systems*. 9881–9893.

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[26] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. 2022. Pretrained transformers as universal computation engines. In *Proceedings of the AAAI conference on Artificial Intelligence*. 7628–7636.

[27] Jun Ma and Bo Wang. 2023. Segment anything in medical images. *arXiv preprint arXiv:2304.12306* (2023).

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

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

[30] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* (2020), 5485–5551.

[31] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[33] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language model with self generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. 13484–13508.

[34] Wei Wei, Chao Huang, Lianghao Xia, and Chuxu Zhang. 2023. Multi-Modal Self-Supervised Learning for Recommendation. In *Proceedings of the ACM Web Conference 2023*. 790–800.

[35] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2022. Transformers in time series: A survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 6778–6786.

[36] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. 2022. Etsformer: Exponential smoothing transformers for time-series forecasting. In *arXiv preprint arXiv:2202.01381*.

[37] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*.

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

[39] Wentao Xu, Weiqing Liu, Chang Xu, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2021. Rest: Relational event-driven stock trend forecasting. In *Proceedings of the Web Conference 2021*. 1–10.

[40] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *Proceedings of the AAAI conference on artificial intelligence*. 11121–11128.

[41] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as instruction following: A large language model empowered recommendation approach. *arXiv preprint arXiv:2305.07001* (2023).

[42] Yunhao Zhang and Junchi Yan. 2022. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*.

[43] Haiteng Zhao, Shengchao Liu, Chang Ma, Hannan Xu, Jie Fu, Zhi-Hong Deng, Lingpeng Kong, and Qi Liu. 2023. GIMLET: A Unified Graph-Text Model for Instruction-Based Molecule Zero-Shot Learning. In *Advances in neural information processing systems*.

[44] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on Artificial Intelligence*. 11106–11115.

[45] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*. 27268–27286.

[46] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. 2023. One Fits All: Power General Time Series Analysis by Pretrained LM. In *Advances in Neural Information Processing Systems*.

## A MORE DISCUSSION ON LANGUAGE MODELS EMPOWERED TIME SERIES FORECASTING

The emergence of large language models, such as GPT-4 and Llama, has swept through and driven advancements in various interdisciplinary fields. For example, the utilization of language models for time series forecasting tasks has recently become a notable research focus [15, 16, 46].

While text data and time series data represent two modalities with noticeable differences, they exhibit inherent similarities that stem from their sequential nature. For instance, both the tasks of language modeling and time series forecasting involve the sequential analysis of data, with the goal of discerning patterns and predicting future elements based on historical observations. Moreover, both text and time series analysis emphasize the importance of recency. In these sequential tasks, recent data holds a heightened relevance, offering more current insights into the evolving patterns and trends. Technically, while language models like GPT2 are typically pretrained on extensive text corpora, they have demonstrated effectiveness in pattern recognition and reasoning over complex sequences of numeric tokens [8, 26]. This capability can be well extended to time series data, as evidenced in this study and in recent concurrent research like GPT4TS [46] and LLMTime [8].

In this work, we advocate a universal forecasting paradigm, which has important implications in real-world scenarios. Consider, for instance, workload forecasting in cloud computing, where a single cloud provider often manages hundreds of time series workloads, each exhibiting diverse data characteristics and differing lengths. The challenge in this context is that it becomes infeasible to train or tune a model for each individual time series due to the vast number of workloads. Our proposed paradigm is devoted to handle this level of complexity, i.e., it provides a more generalized forecasting solution that is crucial when dealing with the practical constraints of managing and predicting in such scenario.

Referring to Table 2 in the main paper, we empirically show that the benefits of universal forecasting also manifest in knowledge sharing or transferring across datasets, as evidenced by overall performance enhancements. The proposed method UniTime improves 37 out of 80 entries to the new state-of-the-art in comparison to baseline models trained separately on each dataset. It also showcases performance improvements during zero-shot transfers.

The recent method, GPT4TS, cannot be regarded as a unified model for cross-domain time series forecasting. As per their official code available at this link: https://github.com/DAMO-DI-ML/NeurIPS2023-One-Fits-All/tree/main/Long-term_Forecasting/scripts, they train a dedicated model for each individual dataset, and even for each predictive length. To evaluate the cross-domain capability of GPT4TS, we modify the implementation of GPT4TS to make it support variable input and output lengths. And according to our empirical results, GPT4TS performs admirably when trained individually on each dataset. However, they encounter difficulties when trained across datasets, experiencing a significant drop in performance on the ETTm1 and Illness datasets. Another relevant method LLMTime [8] can be considered as a unified model due to its direct applicability across different datasets. However, the reported performance in their paper falls short of being competitive.

## B TRAINING CONFIGURATIONS

Table 6 offers detailed configurations for each dataset evaluated in this study. First, we partition all datasets into training, validation and test set in chronological order. The split ratio is 6:2:2 for the ETT series dataset and 7:1:2 for others. We can observe that the datasets ETTm1, ETTm2, and Weather have the highest number of training samples, each exceeding 30,000. They are followed by ETTh1 and ETTh2 with approximately 8,500 samples, Exchange with 5,000 samples, and Illness, which has only 600 samples. Then we determine the batch size for each dataset based on the number of training samples.

The guiding principle is to allocate a larger batch size for datasets with more training samples and a smaller batch size for those with fewer samples. This strategy allows the model to undergo more frequent updates when training on smaller datasets during each epoch. Following this principle, we assign a batch size of 64 to the ETTm1, ETTm2, and Weather datasets, 32 to ETTh1 and ETTh2, 24 to the Exchange dataset, and 16 to the Illness dataset. An exception to this principle is for the Electricity dataset, which is supposed to be set to 32, but due to GPU memory constraints, it is set to 24 in our experiments.

Furthermore, recall that we implement oversampling to augment the size of datasets with significantly fewer training samples. This strategy is applied to the Illness dataset, which contains only 600 samples. The decision to perform oversampling 12 times on the Illness dataset is based on our empirical assessments. In short, the primary goal of the two strategies is to ensure that the model obtains ample exposure to the underrepresented domains, preventing them from being marginalized by the more abundant ones.

## C MORE RESULTS FOR VARIANTS OF INSTRUCTIONS

The domain instructions are essentially sentences that describe the data in each domain. The instructions we employed to attain the overall performance are listed in Table 6. In this part, we construct different variants of instructions and aim to investigate how the model behaves in response to changes in the provided instructions.

First, we consider the set of instructions in Table 6 as the baseline and denote it as *Original*. Subsequently, we generate instruction variants through three types of rephrasing: (1) *Short*: we shorten the original instructions. (2) *Expand*: we expand the original instructions with some general descriptions. (3) *Detail*: we expand the original instructions with additional information about the datasets. We realize the above rephrasing by providing prompts to ChatGPT-3.5-turbo. The prompts are listed in Table 7. To aid comprehension, we also provide two specific examples in the table illustrating how the instructions appear after modification by ChatGPT. We then proceed to randomly generate 5 sets of instructions (labeled from random 1 to random 5). During the generation of each instruction set, we randomly select an instruction variant for each domain.

The first question we explore is *whether our method can accommodate these instructions of varying lengths without fine-tuning the language model*. To this end, we consider the following two training settings: (1) *w/o fine-tuning*: we freeze the language model, and tune the other parts of UniTime. (2) *w/ fine-tuning*: we do not freeze any parameters of UniTime. Please note that we consistently input both

**Table 6: Details of the training, validation, and testing set partitions, as well as the configurations specific to different domains.**

| Dataset | #Training | #Validation | #Testing | Batch Size | Oversample Times | Stride | Domain Instructions |
|---|---|---|---|---|---|---|---|
| ETTm1 | 34,465 | 11,521 | 11,521 | 64 | 0 | 16 | Electricity transformer A data with fifteen minutes sample rate. |
| ETTm2 | 34,465 | 11,521 | 11,521 | 64 | 0 | 16 | Electricity transformer B data with fifteen minutes sample rate. |
| ETTh1 | 8,545 | 2,881 | 2,881 | 32 | 0 | 16 | Electricity transformer A data with one hour sample rate. |
| ETTh2 | 8,545 | 2,881 | 2,881 | 32 | 0 | 16 | Electricity transformer B data with one hour sample rate. |
| Electricity | 18,317 | 2,633 | 5,261 | 24 | 0 | 16 | Power consumption data with hourly sample rate. |
| Weather | 36,792 | 5,271 | 10,540 | 64 | 0 | 16 | Meteorological indicator data with ten minutes sample rate. |
| Exchange | 5,120 | 665 | 1,422 | 24 | 0 | 16 | Exchange rate data with one day sample rate. |
| Illness | 617 | 74 | 170 | 16 | 12 | 4 | Patient number data with one week sample rate. |

**Table 7: Variants of domain instructions.**

| Variants | Prompts for ChatGPT | Example 1 | Example 2 |
|---|---|---|---|
| Original | – | meteorological indicator data with ten minute sample rate. | exchange rate data with one day sample rate. |
| Short | Rephrase the following text shorter: {instruction}. | ten-minute meteorological data. | daily exchange rate data. |
| Expand | Rephrase the following text longer: {instruction}. | the dataset for meteorological indicators presents detailed information, with data points collected at specific ten-minute intervals, facilitating a thorough analysis of meteorological conditions and trends over time. | the dataset for exchange rates provides comprehensive information, with data points recorded at consistent one-day intervals, enabling a detailed examination of currency fluctuations and trends over time. |
| Detail | Rephrase the following text: {instruction}, by adding the information: {information}. | the dataset includes meteorological indicators sampled every ten minutes, collected in the year 2020, and features information on 21 meteorological indicators, including temperature and humidity. | the dataset comprises exchange rate data sampled on a daily basis, documenting the daily exchange rates of eight distinct countries spanning the period from 1990 to 2016. |

**Table 8: Test results for variants of instructions. We report the average values over all predictive lengths.**

| Variants | ETTm1 | | ETTm2 | | ETTh1 | | ETTh2 | | Electricity | | Weather | | Exchange | | Illness | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| w/o fine-tuning random 1 | 0.394 | 0.407 | 0.297 | 0.341 | 0.450 | 0.456 | 0.375 | 0.402 | 0.218 | 0.308 | 0.258 | 0.281 | 0.396 | 0.424 | 2.336 | 0.999 |
| w/ fine-tuning random 1 | 0.393 | 0.404 | 0.295 | 0.338 | 0.446 | 0.452 | 0.381 | 0.408 | 0.211 | 0.301 | 0.257 | 0.279 | 0.381 | 0.417 | 2.291 | 0.957 |
| w/o fine-tuning random 2 | 0.389 | 0.403 | 0.296 | 0.339 | 0.441 | 0.446 | 0.372 | 0.402 | 0.219 | 0.311 | 0.260 | 0.282 | 0.388 | 0.422 | 2.569 | 1.065 |
| w/ fine-tuning random 2 | 0.386 | 0.403 | 0.299 | 0.340 | 0.440 | 0.446 | 0.385 | 0.409 | 0.210 | 0.299 | 0.256 | 0.279 | 0.375 | 0.410 | 2.240 | 0.934 |
| w/o fine-tuning random 3 | 0.396 | 0.406 | 0.295 | 0.339 | 0.454 | 0.460 | 0.374 | 0.405 | 0.220 | 0.314 | 0.260 | 0.282 | 0.388 | 0.420 | 2.278 | 0.975 |
| w/ fine-tuning random 3 | 0.390 | 0.400 | 0.288 | 0.333 | 0.445 | 0.453 | 0.383 | 0.408 | 0.213 | 0.303 | 0.258 | 0.281 | 0.383 | 0.420 | 2.135 | 0.935 |
| w/o fine-tuning random 4 | 0.394 | 0.406 | 0.303 | 0.346 | 0.447 | 0.457 | 0.374 | 0.405 | 0.219 | 0.310 | 0.259 | 0.283 | 0.392 | 0.423 | 2.520 | 1.066 |
| w/ fine-tuning random 4 | 0.386 | 0.405 | 0.304 | 0.344 | 0.446 | 0.453 | 0.386 | 0.411 | 0.209 | 0.298 | 0.257 | 0.280 | 0.372 | 0.412 | 2.074 | 0.897 |
| w/o fine-tuning random 5 | 0.394 | 0.405 | 0.295 | 0.340 | 0.439 | 0.449 | 0.379 | 0.409 | 0.218 | 0.309 | 0.263 | 0.285 | 0.381 | 0.416 | 2.444 | 1.016 |
| w/ fine-tuning random 5 | 0.390 | 0.401 | 0.293 | 0.336 | 0.441 | 0.450 | 0.387 | 0.411 | 0.212 | 0.300 | 0.259 | 0.281 | 0.380 | 0.416 | 2.239 | 0.944 |

the instructions and time series data into the model, regardless of whether the language model is frozen or not.

The experimental results are presented in Table 8. We observe that, across all five sets of instructions with varying lengths, while the performance of *w/ fine-tuning* generally surpasses *w/o fine-tuning*, the discrepancies in performance are not notably significant. The intuition here is that even when the language model is frozen, the instruction (no matter if it is short or long) remains an unchanged signal for each domain. Consequently, the model retains the ability to distinguish domains and achieve reasonable results without ensuring homogeneity of instructions.

Second, by examining the five random results obtained when fine-tuning the language model, we can derive conclusions regarding *the stability of our method under various instruction rephrasing*. We observe that, across all instruction rephrasing sets, the discrepancies

in performance are not notably significant. These results suggest that our method exhibits a degree of robustness or stability to the tested approaches of rephrasing. This implication further suggests that instructions may not necessarily require meticulous crafting, and adopting simple approaches, such as using large language models to generate instructions based on the meta-data of a specific dataset, may not significantly compromise performance.

## D MORE RESULTS FOR UNITIME BACKBONE

In this part, we conduct experiments to assess the efficacy of employing GPT-2 or T5 as the underlying architecture for our UniTime model. The results are presented in Table 9, which indicate that T5 does not surpass GPT-2 in terms of performance. This discrepancy may be attributed to GPT-2's utilization of causal masking, which preserves the temporal order of inputs – potentially crucial

**Table 9: UniTime backbone using GPT2 vs T5. We report the average results over all predictive lengths.**

| Variants | ETTm1 | | ETTm2 | | ETTh1 | | ETTh2 | | Electricity | | Weather | | Exchange | | Illness | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| UniTime w/ T5 | 0.397 | 0.408 | 0.300 | 0.339 | 0.450 | 0.453 | 0.390 | 0.412 | 0.227 | 0.316 | 0.263 | 0.283 | 0.379 | 0.415 | 2.210 | 0.932 |
| UniTime w/ GPT2 | 0.385 | 0.399 | 0.293 | 0.334 | 0.442 | 0.448 | 0.378 | 0.403 | 0.216 | 0.305 | 0.253 | 0.276 | 0.364 | 0.404 | 2.137 | 0.929 |
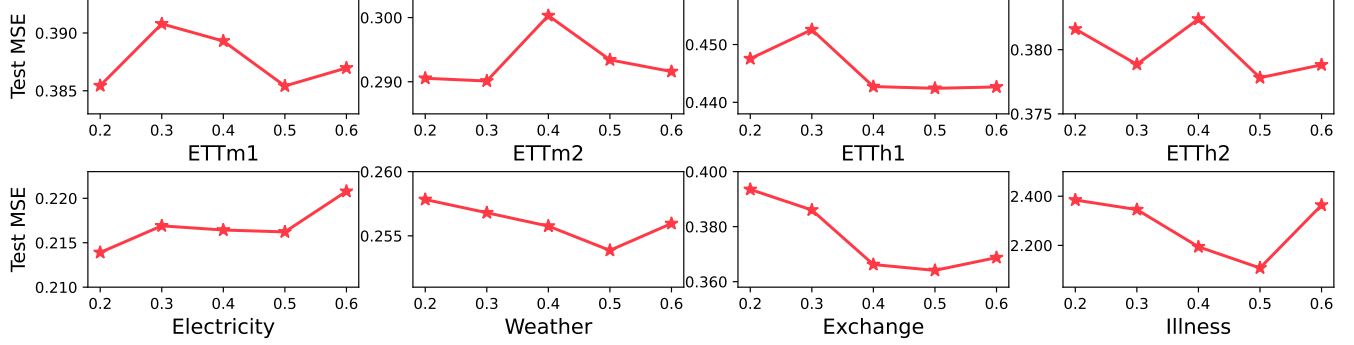


Figure 5: Effects of mask ratio. The y-axis is the average test MSE over four predictive lengths.
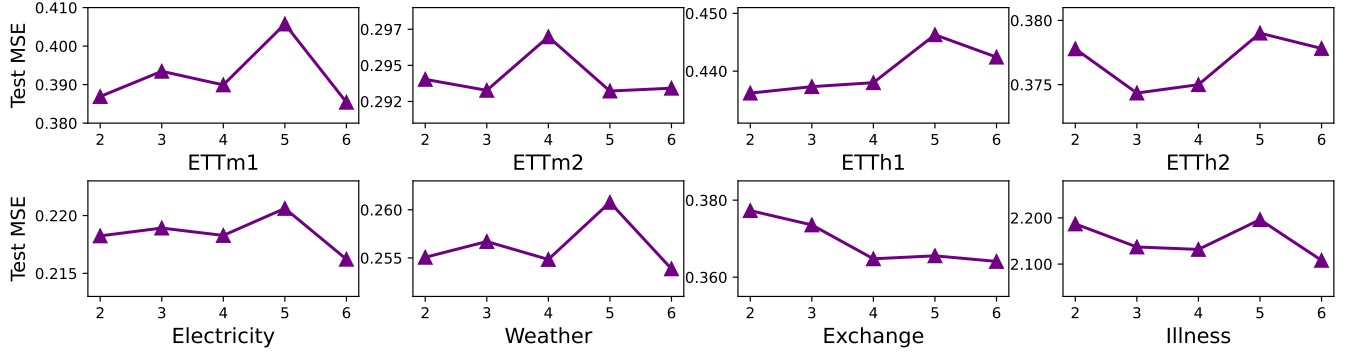


Figure 6: Effects of Language-TS Transformer's number of layers. The y-axis is the average test MSE over four predictive lengths.

for both textual and time series data. On the contrary, T5 employs bidirectional attention mechanisms.

# E  HYPERPARAMETER STUDIES

In this part, we conduct an investigation into two critical hyperparameters: the mask ratio $r_m$ and the number of layers $L_{lm}$ in the Language-TS Transformer. The results of these assessments are depicted in Figure 5 and Figure 6. Regarding the mask ratio value, we observe that the model generally performs better with a larger ratio compared to a smaller one. The best performance is generally obtained when the ratio is set to 0.5. As for the number of layers in the Language-TS Transformer, a count of 6 appears to be the most favorable choice. We refrain from setting the number of layers to a larger value, such as 7, due to constraints imposed by limitations in GPU memory.