ROSE: REGISTER-ASSISTED GENERAL TIME SERIES FORECASTING WITH DECOMPOSED FREQUENCY LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

With the increasing collection of time series data from various domains, there arises a strong demand for general time series forecasting models pre-trained on a large number of time-series datasets to support a variety of downstream prediction tasks. Enabling general time series forecasting faces two challenges: how to obtain unified representations from multi-domian time series data, and how to capture domain-specific features from time series data across various domains for adaptive transfer in downstream tasks. To address these challenges, we propose a **R**egister Assisted General Time Series F**o**recasting Model with Decompo**se**d Frequency Learning (**ROSE**), a novel pre-trained model for time series forecasting. ROSE employs Decomposed Frequency Learning for the pre-training task, which decomposes coupled semantic information in time series with frequency-based masking and reconstruction to obtain unified representations across domains. We also equip ROSE with a Time Series Register, which learns to generate a register to capture domain-specific representations during pre-training and enhances domain-adaptive transfer by selecting related register tokens on downstream tasks. After pre-training on large-scale time series data, ROSE achieves state-of-the-art forecasting performance on 7 real-world benchmarks. Remarkably, it demonstrates competitive or superior few-shot and zero-shot abilities.

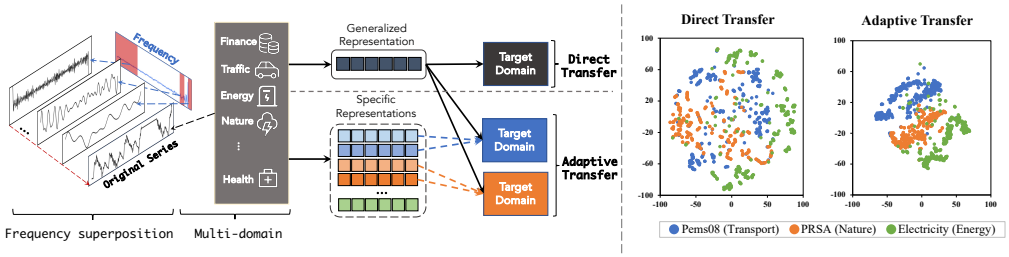
1 INTRODUCTION

Time Series Forecasting plays a pivotal role across numerous domains, such as energy, smart transportation, weather, and economics (Qiu et al., 2024). However, training specific models in deep learning for each dataset is costly and requires tailored parameter tuning, whose prediction accuracy may be limited due to data scarcity (Liu et al., 2024). One solution is to pre-train a general model on diverse time series datasets and fine-tune it with a few data for different downstream scenarios or direct predict without fine-tuning. Following this idea, foundation models for time series forecasting have recently raised growing attention, continually scaling up the pre-training datasets and model sizes to improve the generalization performance (Woo et al., 2024; Goswami et al., 2024; Ansari et al., 2024). However, an excessively large scale also causes increasing costs in training and inference, which may go against the original intention of a general model, especially in resource-constrained situations. In addition to scaling up, general time series forecasting models can also be designed from the perspectives of pre-training tasks and downstream transfer adaptation. From these two angles, we identify the following challenges.

Obtaining a unified representation from time series data across various domains is challenging. Time series from each domain involve complex temporal patterns, composed of multiple frequency components combined with each other (Zhou et al., 2022), which is frequency superposition. As shown in Figure 1(a), different frequency components contain distinct semantic information. For example, low and high-frequency components represent long-term trends and rapid variations, respectively (Zhang et al., 2022). Furthermore, different datasets exhibit diverse frequency distributions, and the significance of low-frequency and high-frequency components for time series modeling varies across domains(Zhang et al., 2024). As a result, large-scale time series data from different domains introduce even more complex temporal patterns and frequency diversity. Existing pre-training frameworks (Dong et al., 2024; Nie et al., 2022; Lee et al., 2023), such as masked modeling and

contrastive learning, were proposed to learn a unified representation from time domain. However, these methods overlook the frequency diversity and complexity exhibited in heterogeneous time series that come from various domains, making it difficult to capture intricate patterns, thus limiting their generalization capabilities.

Adaptive transferring information from multi-domain time series to specific downstream scenarios presents a challenge. Multi-source time series data originate from various domains (Woo et al., 2024), whose data exhibit domain-specific information (Liu et al., 2024). Information from the same or similar domain as the target domain is useful for improving the model's effectiveness in the target task (Chen et al., 2023a). However, as shown in Figure 1(a), existing time series pre-training frameworks (Woo et al., 2024; Liu et al., 2024; Zhou et al., 2024) focus mainly on learning generalized representation during pre-training and overlook domain-specific representation. Thus, they only transfer the same generalized representation to different target domains, called *direct transfer*, which limits the model's effectiveness in specific downstream tasks. Therefore, it is necessary to learn domain-specific information during pre-training and adaptively transfer the specific representations to each target domain, called *adaptive transfer*. Realizing adaptive transfer poses two difficulties: 1) capturing domain-specific information in pre-training. 2) adaptive use of domain-specific information in various downstream tasks.



(a) the illustration of general time series forecasting **(b) t-SNE visualization**

Figure 1: **(a)** Pre-training on multi-domain datasets that exhibit combined frequency. Existing general time series forecasting models only extract generalized representations for direct transfer to various downstream target domains. We propose to learn generalized and specific representations during pre-training, and adaptively transfer them to each target domain. **(b)** The t-SNE visualization of the hidden representations after direct transfer and adaptive transfer: In direct transfer, representations of different domains are mixed, but in adaptive transfer, they show a clear clustering pattern. The detailed experiment setting is in the Appendix A.10.4

To address these challenges, we propose a **r**egister assisted general time series f**o**recasting model with decompos**e**d frequency learning (**ROSE**). **First**, we propose Decomposed Frequency Learning that learns generalized representations to solve the issue with coupled semantic information. We decompose individual time series using the Fourier transform with a novel frequency-based masking method, and then convert it back to the time domain to obtain decoupled time series for reconstruction. It makes complex temporal patterns disentangled, thus benefiting the model to learn generalized representations. **Second**, we introduce Time Series Register (TS-Register) to learn domain-specific information in multi-domain data. By setting up a register, we generate register tokens to learn each domain-specific information during pre-training. In a downstream scenario, the model adaptively selects Top-K vectors from the register that are close to the target domain of interest. During fine-tuning, we adjust the selected register tokens with a novel learnable low-rank matrix, which complements target-specific information to perform more flexible adaptive transfer. As shown in Figure 1(b), adaptive transfer successfully utilizes domain-specific information in multi-domain time series, which contributes to the model's performance in target tasks. The contributions are summarized as follows:

- We propose ROSE, a novel light weight general time series forecasting model using multi-domain datasets for pre-training and improving downstream fine-tuning performance and efficiency.
- We propose a novel Decomposed Frequency Learning that employs multi-frequency masking to learn complex general temporal patterns from multi-domain data, empowering the model's generalization capability.

- We propose a novel TS-Register to capture domain-specific information in pre-training and enable adaptive transfer of target-oriented specific information for downstream tasks.
- Our experiments with 7 real-world benchmarks demonstrate that ROSE achieves state-of-the-art performance in full-shot setting and achieves competitive or superior results in few-shot setting, along with impressive transferability in zero-shot setting.

2 Related work

2.1 Traditional time series forecasting

The statistical time series forecasting models like ARIMA (Box and Jenkins, 1968), despite their theoretical support, are limited in modeling nonlinearity. With the rise of deep learning, many RNN-based models (Cirstea et al., 2019; Wen et al., 2017; Salinas et al., 2020) have been proposed, modeling the sequential data with an autoregressive process. CNN-based models (Luo and Wang, 2024; Liu et al., 2022a) have also received widespread attention due to their ability to capture local features. MICN (Wang et al., 2022) utilizes TCN to capture both local and global features, while TimesNet (Wu et al., 2022) focuses on modeling 2D temporal variations. However, both RNNs and CNNs struggle to capture long-term dependencies. Transformer-based models (Zhou et al., 2022; Nie et al., 2022; Wu et al., 2021; Liu et al., 2023; Chen et al., 2024), with their attention mechanism, can capture long dependencies and extract global information, leading to widespread applications in long-time series prediction. However, this case-by-case paradigm requires meticulous hyperparameter design for different datasets, and its predictive performance can also be affected by data scarcity.

2.2 Pre-trained models for times seires forecasting

LLMs for time series forecasting: Recent studies have shown that Large Language Models (LLMs) can enhance time series forecasting with limited fine-tuning (Zhou et al., 2024), prompting (Jin et al., 2024; Cao et al., 2024), and modality alignment (Jin et al., 2024). GPT4TS (Zhou et al., 2024) fine-tunes a subset of LLM parameters, and achieves competitive performance by leveraging pre-trained text knowledge. TimeLLM (Jin et al., 2024) transforms time series into text to align with LLM representations. Despite their positive impact, LLMs' high inference costs limit their practicality.

Time series foundation model: Pre-training with multiple sources time series has recently received widespread attention (Rasul et al., 2023; Dooley et al., 2024; Garza and Mergenthaler-Canseco, 2023; Kamarthi and Prakash, 2023). MOMENT (Goswami et al., 2024) and MOIRAI (Woo et al., 2024) adopt a BERT-style pre-training approach, while Timer (Liu et al., 2024), Chronos (Ansari et al., 2024) and TimsFM (Das et al., 2023a) use a GPT-style pre-training approach, giving rise to improved performance in time series prediction. However, the above methods overlook domain-specific information from multi-source data, thus limiting the performance of the models. Different from previous approaches, ROSE pre-trains on large-scale data from various domains and it considers both generalized representations and domain-specific information, which facilitates flexible adaptive transfer in downstream tasks.

3 Methodology

Problem Definition. Given a multivariate time series $\mathbf{X}_t = \{\mathbf{x}^i_{t-L:t}\}^C_{i=1}$, where each $\mathbf{x}^i_{t-L:t} \in \mathbb{R}^L$ is a sequence of observations. L denotes the look-back window and C denotes the number of channels. The forecasting task is to predict the future values $\hat{\mathbf{Y}}_t = \{\hat{\mathbf{x}}^i_{t:t+F}\}^C_{i=1}$, where F denotes the forecast horizon. $\mathbf{Y}_t = \{\mathbf{x}^i_{t:t+F}\}^C_{i=1}$ is the ground truth of future.

The general time series forecasting model is pre-trained with multi-source datasets $\mathbf{D}_{\text{pre-train}} = \{(\mathbf{X}^j_t, \mathbf{Y}^j_t)\}^N_{j=1}$, where N is the number of datasets. For the downstream task, the model is fine-tuned with a training dataset $\mathbf{D}_{\text{train}} = \{(\mathbf{X}^{\text{train}}_t, \mathbf{Y}^{\text{train}}_t)\}$, and is tested with $\mathbf{D}_{\text{test}} = \{(\mathbf{X}^{\text{test}}_t, \mathbf{Y}^{\text{test}}_t)\}$ to predict $\hat{\mathbf{Y}}^{\text{test}}_t$, where $\mathbf{D}_{\text{pre-train}}$, $\mathbf{D}_{\text{train}}$ and $\mathbf{D}_{\text{test}}$ are pairwise disjoint. Alternatively, the model could be directly tested using $\mathbf{D}_{\text{test}}$ without fine-tuning with $\mathbf{D}_{\text{train}}$ to predict $\hat{\mathbf{Y}}^{\text{test}}_t$.

3.1 ARCHITECTURE

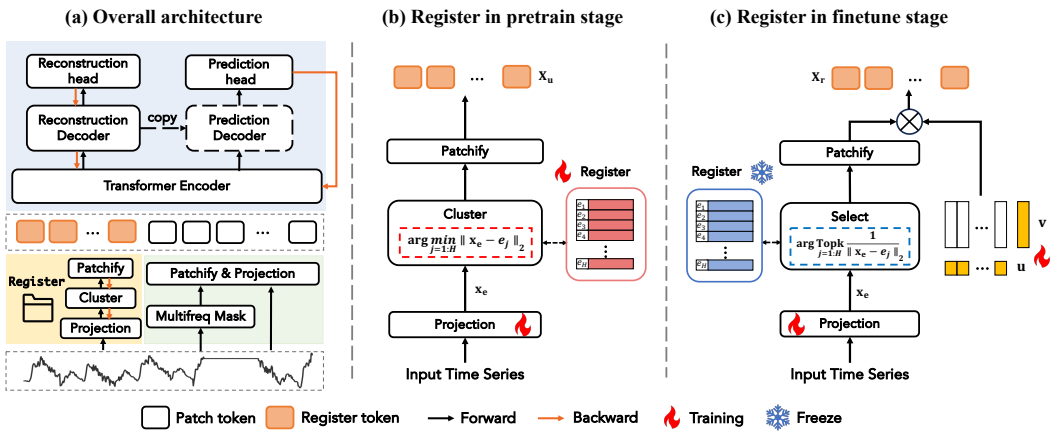


Figure 2: The model architecture of ROSE.

The time series forecasting paradigm of ROSE contains two steps: *pre-training* and *fine-tuning*. First, ROSE is pre-trained on large-scale datasets from various domains, with two pre-training tasks: reconstruction and prediction. We set up the reconstruction task to help the model understand time series comprehensively and set the prediction task to enhance the model's few-shot and zero-shot abilities. Then, ROSE is fine-tuned with a target dataset in the downstream scenario.

As shown in Figure 2, ROSE employs the encoder-decoder architecture to model time series. The backbone consists of multiple Transformer layers to process sequential information and effectively capture temporal dependencies (Vaswani et al., 2017). The reconstruction decoder and prediction decoder use the same structure as the Transformer encoder. They are used for reconstruction and prediction tasks respectively. ROSE is pre-trained in a channel-independent way, which is widely used in time series forecasting (Nie et al., 2022).

Input Representations. To enhance the generalization of ROSE for adaptive transferring from multi-domains to different target domains, we model the inputs $\mathbf{x}$ with **patch tokens** and **register tokens**. Patch tokens are obtained by partitioning the time series using patching layers (Nie et al., 2022), to preserve local temporal information. Register tokens that capture domain-specific information will be introduced in Section 3.3.

3.2 DECOMPOSED FREQUENCY LEARNING

As shown in Figure 1, time series data are composed of multiple superimposed frequency components, resulting in the overlap of different temporal changes. Furthermore, low-frequency components typically contain information about overall trends and longer-scale variations, and high-frequency components usually contain information about short-term fluctuations and shorter-scale variations, therefore, understanding time series from low and high frequencies separately benefits general time series representation learning. Based on the observations above, we propose a novel frequency-based masked modeling that randomly mask either high-frequency or low-frequency components of a time series multiple times as the key to enable learning of common time series patterns, such as trends and various long and short term fluctuations. Finally, reconstruction task assists the model in comprehending the data from various frequency perspectives, enabling it to learn generalized representations. In contrast, existing frequency masking methods (Zhang et al., 2022), which randomly mask frequencies of a single time series once, show limited forecasting effectiveness due to the lack of common pattern learning from heterogeneous time series that come from various domains.

Multi-frequency masking. As shown in the green part of Figure 3, given a time series $\mathbf{x} \in \mathbb{R}^L$, we utilize the Real Fast Fourier Transform (rFFT) (Brigham and Morrow, 1967) to transform it into the frequency domain, giving rise to $\mathbf{x}_{\text{freq}} \in \mathbb{C}^{L/2+1}$.

$$\mathbf{x}_{\text{freq}} = \text{rFFT}(\mathbf{x}). \tag{1}$$

To separately model high-frequency and low-frequency information in time series, we sample K_{f} thresholds $\tau_1, \tau_2, \tau_3, ..., \tau_{K_{\text{f}}}$ and K_{f} random numbers $\mu_1, \mu_2, \mu_3, ..., \mu_{K_{\text{f}}}$ for multi-frequency masks,

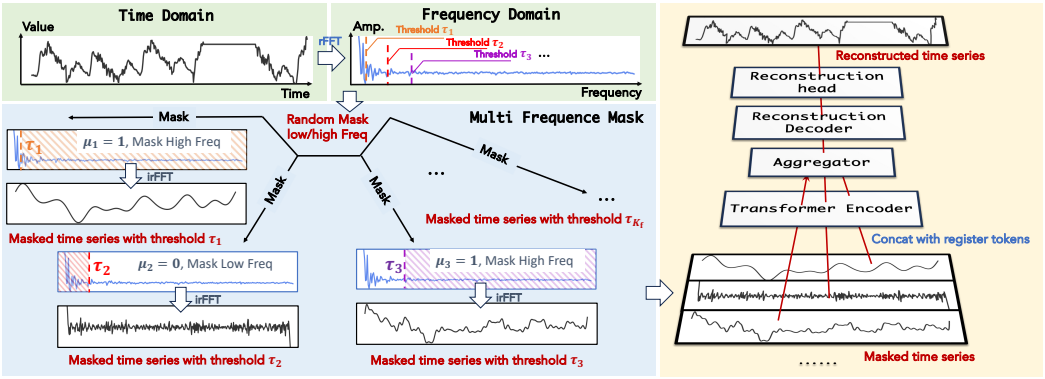


Figure 3: An illustration of decomposed frequency learning. Based on the sampled thresholds, we randomly apply low/high-frequency masking to the time series in the frequency domain and then transform it back to the time domain for reconstruction.

where $\tau \in \text{Uniform}(0, a)$, $a < L/2 + 1$, and $\mu \in \text{Bernoulli}(p)$. Each pair of τ_i and μ_i corresponds to the i_{th} frequency mask. This generates a mask matrix $\mathbf{M} \in \{0, 1\}^{K_\text{f} \times (L/2+1)}$, where each row corresponds to the i_{th} frequency mask, each column corresponds to the j_{th} frequency, and each element m_{ij} is 0 or 1, meaning that the j_{th} frequency is masked with the i_{th} frequency mask or not.

$$m_{ij} = \begin{cases} \mu_i & , if \ j < \tau_i \\ (1 - \mu_i) & , if \ j > \tau_i \end{cases}, \tag{2}$$

where τ_i and μ_i denote the threshold and random number for the i_{th} frequency domain mask. If $\mu_i = 1$, it means that frequency components above τ_i will be masked, indicating to mask high frequency , as shown by the threshold τ_1 in Figure 3. Conversely, if $\mu_i = 0$, it signifies that frequency components below τ_i will be masked, indicating to mask low frequency, exemplified by threshold τ_2 in Figure 3.

After obtaining the mask matrix $\mathbf{M}$, we replicate $\mathbf{x}_\text{freq}$ K_f times to get the $\mathbf{X}_\text{freq} \in \mathbb{C}^{K_\text{f} \times L/2+1}$ and perform element-wise Hadamard product with the mask matrix $\mathbf{M}$ to get masked frequency of time series. Then, we use the inverse Real Fast Fourier Transform (irFFT) to convert the results from the frequency domain back to the time domain and get K_f masked sequences $\mathbf{X}_\text{mask} = \{\mathbf{x}^i_\text{mask}\}^{K_\text{f}}_{i=1}$, where each $\mathbf{x}^i_\text{mask} \in \mathbb{R}^L$ corresponding to masking with a different threshold τ_i.

$$\mathbf{X}_\text{mask} = \text{irFFT}(\mathbf{X}_\text{freq} \odot \mathbf{M}). \tag{3}$$

Representation learning. As shown in the yellow part of Figure 3, after obtaining the K_f masked sequences $\mathbf{X}_\text{mask}$, we divide each sequence $\mathbf{x}^i_\text{mask}$ into P non-overlapping patches, and use a linear layer to transforming them into P patch tokens, and thus we get $\mathcal{X}_\text{mp} = \{\mathbf{X}^i_\text{mp}\}^{K_\text{f}}_{i=1}$ to capture general information, where each $\mathbf{X}^i_\text{mp} \in \mathbb{R}^{P \times D}$, and D is the dimension for each patch token. We replicate the register tokens $\mathbf{X}_\text{u}$ K_f times to get $\mathcal{X}_\text{u} \in \mathbb{R}^{K_\text{f} \times N_\text{r} \times D}$, where $\mathbf{X}_\text{u} \in \mathbb{R}^{N_\text{r} \times D}$ is obtained by inputting the original sequence into the TS-Register, as detailed in Section 3.3. Then, we concatenate the patch tokens $\mathcal{X}_\text{mp}$ with the register tokens $\mathcal{X}_\text{u}$, and feed them into the Transformer encoder to obtain the representation of each masked series. These representations are then aggregated to yield a unified representation $\mathbf{S}_\text{m} \in \mathbb{R}^{(N_\text{r}+P) \times D}$. The aggregator is the averaging operation.

$$\mathbf{S}_\text{m} = \text{Aggregator}(\text{Encoder}(\text{Concatenate}(\mathcal{X}_\text{mp}, \mathcal{X}_\text{u}))). \tag{4}$$

Reconstruction task. After obtaining the representation $\mathbf{S}_\text{m}$, we feed it into the reconstruction decoder, which shares same stucture as the Tranformer encoder, and ultimately reconstruct the original sequence $\hat{\mathbf{x}} \in \mathbb{R}^L$ through the reconstruction head, which is a linear layer. As frequency domain masking affects the overall time series, we compute the Mean Squared Error (MSE) reconstruction loss for the entire time series.

$$\mathcal{L}_\text{reconstruction} = ||\mathbf{x} - \hat{\mathbf{x}}||^2_2. \tag{5}$$

3.3 Time Series Register

By decomposed frequency learning, we can obtain the general representations, and additionally, we propose the TS-Register that learns register tokens as the domain-specific information from the

multi-domain datasets for adaptive transfer. It clusters domain-specific information from the multi-domain datasets into register tokens and stores such domain-specific information in the register during pre-training. Then, it adaptively selects domain-specific information from the register via a Top-K selection strategy to enhance the performance in the target domain. A novel learnable low-rank matrix is proposed to set to complement the downstream dataset-specific information through fine-tuning.

We set up a randomly initialized register $\mathbf{E} \in \mathbb{R}^{H \times D_{\mathrm{r}}}$ with H cluster center vectors $\mathbf{e}_i \in \mathbb{R}^{D_{\mathrm{r}}}, i \in \{1, 2, \ldots, H\}$. Each of input time series $\mathbf{x} \in \mathbb{R}^{L}$ is projected into a data-dependent embedding $\mathbf{x}_{\mathrm{e}} \in \mathbb{R}^{D_{\mathrm{r}}}$ through a linear layer.

Pre-training stage. As shown in Figure 2(b), we use the register to cluster these data-dependent embeddings, which generate domain-specific information, and store them in pre-training. Specifically, We find a cluster center vector $\mathbf{e}_\delta$ from the register $\mathbf{E}$ where we use δ to denote the cluster that the data-dependent embedding $\mathbf{x}_{\mathrm{e}}$ belongs to.

$$\mathcal{L}_{\text{register}} = \|\mathbf{x}_{\mathrm{e}} - \mathbf{e}_\delta\|_2^2, \;\; \delta = \underset{j=1:H}{\arg\min} \|\mathbf{x}_{\mathrm{e}} - \mathbf{e}_j\|_2 \,. \tag{6}$$

To update the cluster center vectors in the register $\mathbf{E}$ that represents the domain information of the pre-trained datasets, we set the loss function shown in Equation 6 that minimizes the distance between the data-dependent embedding $\mathbf{x}_{\mathrm{e}}$ and the cluster center $\mathbf{e}_\delta$. To solve the problem that the gradient of the $\arg\min$ function cannot be backpropagated, we use the stop gradient operation to pass the gradient of $\mathbf{e}_\delta$ directly to $\mathbf{x}_{\mathrm{e}}$.

In this way, the vectors in the register $\mathbf{E}$ cluster the embeddings of different data and learn the domain-specific centers for pre-trained datasets, which can represent domain-specific information. As a vector in the register $\mathbf{E}$, $\mathbf{e}_\delta$ represents the domain-specific information for input $\mathbf{x}$. $\mathbf{e}_\delta$ is invariant under small perturbations in $\mathbf{x}_{\mathrm{e}}$ that represents $\mathbf{x}$, which promotes better representation of domain-specific information and robustness of the vectors in the register. This also avoids their over-reliance on detailed information about specific datasets.

The cluster center vector $\mathbf{e}_\delta$ is then patched into $\mathbf{X}_{\mathrm{u}} \in \mathbb{R}^{N_{\mathrm{r}} \times D}$, where N_{r} is the number of the register tokens and D is the dimensionality of Transformer latent space. $\mathbf{X}_{\mathrm{u}}$ is called register tokens, which are used as the prefix of the patch tokens $\mathbf{X}_{\mathrm{p}} \in \mathbb{R}^{P \times D}$ and input for the Transformer encoder to provide domain-specific information.

Fine-tuning stage. As shown in Figure 2(c), after obtaining a register $\mathbf{E}$ that contains domain-specific information through pre-training, we freeze the register parameters to adaptively use this domain-specific information in the downstream tasks.

Since the target domain may not strictly fit one of the upstream domains, we propose a novel embedding learning of the downstream data by employing a Top-K strategy that selects k similar vectors in the register. As shown in Equation 7, the embedding of input time series $\mathbf{x}_{\mathrm{e}}$ picks the k nearest vectors in the register $\mathbf{E}$, and uses their average as $\bar{\mathbf{e}}_k$ to represent the domain-specific information from the pre-train stage. $\bar{\mathbf{e}}_k$ is also patched into $\mathbf{X}_{\mathrm{d}} \in \mathbb{R}^{N_{\mathrm{r}} \times D}$ and is used as the **domain specific register tokens**.

$$\bar{\mathbf{e}}_k = \frac{1}{k} \sum_{i=1}^{k} \mathbf{e}_{\delta_i}, \;\; \{\delta_1, \cdots, \delta_k\} = \underset{j=1:H}{\arg\operatorname{Topk}}\left(\frac{1}{\|\mathbf{x}_{\mathrm{e}} - \mathbf{e}_j\|_2}\right). \tag{7}$$

Since the downstream data has its own specific information at the dataset level in addition to the domain level, this may not be fully represented by the domain information obtained from the pre-trained dataset alone. Therefore, we innovatively set a learnable matrix $\mathbf{A} \in \mathbb{R}^{N_{\mathrm{r}} \times D}$ to adjust $\mathbf{X}_{\mathrm{d}}$ to complement the **specific information of downstream data**. Since the pre-trained model has a very low intrinsic dimension (Aghajanyan et al., 2020), in order to get better fine-tuning results, $\mathbf{A}$ is set as a low-rank matrix:

$$\mathbf{A} = \mathbf{u} \times \mathbf{v}^{\mathrm{T}}, \tag{8}$$

where $\mathbf{u} \in \mathbb{R}^{N_{\mathrm{r}}}$ and $\mathbf{v} \in \mathbb{R}^{D}$, and only the vectors $\mathbf{u}$ and $\mathbf{v}$ need to be retrained in the fine-tuning step. As illustrated in Equation 9, the register token $\mathbf{X}_{\mathrm{r}}$ of the downstream scenario is obtained by doing the Hadamard product of $\mathbf{X}_{\mathrm{d}}$, which represents the domain-specific information obtained at the pre-train stage, and $\mathbf{A}$, which represents the downstream dataset-specific information.

$$\mathbf{X}_{\mathrm{r}} = \mathbf{X}_{\mathrm{d}} \odot \mathbf{A}. \tag{9}$$

3.4 TRAINING

To improve the model's prediction ability in zero-shot and few-shot setting, we co-train supervised prediction with self-supervised reconstruction that uses multi-frequency masking to learn unified features that are more applicable to the downstream prediction task. We normalize time series by employing the REVIN (Kim et al., 2021) that is commonly used by the state-of-the-art time series models, which first normalizes each input sample and subsequently applies inverse normalization to recover the model's output.

Prediction task. The input time series $\mathbf{x} \in \mathbb{R}^L$ is sliced into P non-overlapping patches and then mapped to $\mathbf{X}_{\mathrm{p}} \in \mathbb{R}^{P \times D}$. Based on common forecasted needs (Qiu et al., 2024), we set up four prediction heads mapping to prediction lengths of $\{96, 192, 336, 720\}$ to accomplish the prediction task. Patch tokens $\mathbf{X}_{\mathrm{p}}$ are concatenated with the register tokens $\mathbf{X}_{\mathrm{u}}$ and then successively fed into the Transformer encoder to yield the representation $\mathbf{S} \in \mathbb{R}^{(N_{\mathrm{r}}+P) \times D}$:

$$\mathbf{S} = \mathrm{Encoder}(\mathrm{Concatenate}(\mathcal{X}_{\mathrm{p}}, \mathcal{X}_{\mathrm{u}})). \tag{10}$$

We feed the representation $\mathbf{S}$ into the prediction decoder and prediction heads to obtain four prediction results $\hat{\mathbf{Y}}_F$, where $F \in \{96, 192, 336, 720\}$. With the ground truth $\mathbf{Y}_F$, the prediction loss $\mathcal{L}_{\text{prediction}}$ is shown in Equation 11.

$$\mathcal{L}_{\text{prediction}} = \sum_{F \in \{96,192,336,720\}} ||\mathbf{Y}_F - \hat{\mathbf{Y}}_F||_2^2. \tag{11}$$

Pre-training. The reconstruction task learns generalized features through the Transformer encoder and reconstruction decoder. To utilize these features for the prediction task, the parameters of the reconstruction decoder are copied to the prediction decoder during forward propagation. To avoid prediction training affecting the generalization performance of the model, the gradients of the prediction heads are skipped at back-propagation. The overall loss of ROSE in pre-training stage is shown in Equation 12.

$$\mathcal{L}_{\text{pre-train}} = \mathcal{L}_{\text{reconstruction}} + \mathcal{L}_{\text{prediction}} + \mathcal{L}_{\text{register}}. \tag{12}$$

Fine-tuning. We only perform a prediction task in fine-tuning. Patch tokens $\mathbf{X}_{\mathrm{p}}$ are concatenated with the adjusted register tokens $\mathbf{X}_{\mathrm{r}}$. For a downstream task with a fixed prediction length, we use the corresponding pre-trained prediction head to fine-tune the model.

4 EXPERIMENTS

Pre-training datasets. The datasets are crucial for pre-training a general time series forecasting model. In light of this, we gather a considerable amount of publicly available datasets from various domains, including energy, nature, health, transport, web, and economics, etc. The details of these datasets are shown in the Appendix A.1.1. To enhance data utilization, we downsample fine-grained datasets to coarser granularity, resulting in approximately 887 million time points.

Evaluation datasets. To conduct comprehensive and fair comparisons for different models, we conducted experiments on seven well-known forecasting benchmarks as the target datasets, including Weather, Traffic, Electricity, and ETT (4 subsets), which cover multiple domains.

Baselines. We select the state-of-the-art models as our baselines in full-shot and few-shot setting, including four specific models: iTransformer (Liu et al., 2023), PatchTST (Nie et al., 2022), TimesNet (Wu et al., 2022), and DLinear (Zeng et al., 2023), and two LLM-based models: GPT4TS (Zhou et al., 2024) and $\mathbf{S}^2$IP-LLM (Pan et al., 2024). In addition, we select five foundation models for comparison in zero-shot setting, including Timer (Liu et al., 2024), MOIRAI (Woo et al., 2024), Chronos (Ansari et al., 2024), TimesFM (Das et al., 2023b), and Moment (Goswami et al., 2024).

Setup. Consistent with previous works, we adopted Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics. For fair comparison, all methods fix the look-back window L = 512 and predict the future values with lengths $F = \{96, 192, 336, 720\}$. More implementation details are presented in the Appendix A.1.3.

4.1 IN DISTRIBUTION FORECASTING

Setting. In full-shot setting, we fine-tune pre-trained ROSE and the baselines with the full downstream data. In few-shot setting, we fine-tune all models with only 10% train data.

Full-shot results. As shown in Table 1, we also present the results of the ROSE in 10% few-shot setting. Key observations are summarized as follows. First, as a general forecasting model, ROSE ***achieves superior performance compared to the six state-of-the-art baselines with full-data training, achieving an average MSE reduction of 15%,*** which shows that our decomposed frequency learning and register help to learn generalized representations from large-scale datasets and adaptively transfer the multi-domain information to specific downstream scenarios. Second, we observe that ROSE in 10% few-shot setting shockingly ***improves a large margin as MSE reduction in average exceeding 12% over the baselines trained with full data.*** This observation validates the transferability of ROSE pre-trained with large multi-source data.

Table 1: The results for ROSE in full-shot setting and 10% few-shot setting, compared with other methods in full-shot setting. The average results of all predicted lengths are listed here.

Models	ROSE		ROSE (10%)		ITransformer		PatchTST		Timesnet		Dlinear		GPT4TS		S^2IP-LLM	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.391**	**0.414**	0.397	0.419	0.439	0.448	0.413	0.434	0.582	0.533	0.416	0.436	0.427	0.426	0.406	0.427
ETTh2	**0.331**	**0.374**	0.335	0.380	0.374	0.406	**0.331**	0.381	0.409	0.438	0.508	0.485	0.354	0.394	0.347	0.391
ETTm1	**0.341**	**0.367**	0.349	0.372	0.362	0.391	0.353	0.382	0.490	0.464	0.356	0.378	0.352	0.383	0.343	0.379
ETTm2	**0.246**	**0.305**	0.250	0.308	0.269	0.329	0.256	0.317	0.317	0.358	0.259	0.325	0.266	0.326	0.257	0.319
Weather	**0.217**	**0.251**	0.224	0.252	0.233	0.271	0.226	0.264	0.329	0.336	0.239	0.289	0.237	0.270	0.222	0.259
Electricity	**0.155**	**0.248**	0.164	0.253	0.164	0.261	0.159	0.253	0.195	0.296	0.166	0.267	0.167	0.263	0.161	0.257
Traffic	**0.390**	**0.264**	0.418	0.278	0.397	0.282	0.391	**0.264**	0.623	0.333	0.433	0.305	0.414	0.294	0.405	0.286

Few-shot results. The results under the 10% few-shot setting are presented in Table 13 in Appendix A.10.2. ROSE outperforms advanced models when training data is scarce in the target domain. Figure 4 shows the performance of pre-trained ROSE and ROSE trained from scratch on ETTh1 and ETTm2 with different fine-tuning data percentages, noting the best baselines in full-shot setting. The pre-trained ROSE shows stable, superior performance even with limited fine-tuning samples. Specifically, the pre-trained ROSE ***exceeds SOTA performance with only 1% train data for ETTh1 and 2% for ETTm2.*** Moreover, compared to the ROSE trained from scratch, the pre-trained ROSE exhibits a slower decline in predictive performance with the reduction of fine-tuning data, demonstrating the impressive generalization ability of ROSE through pre-training.

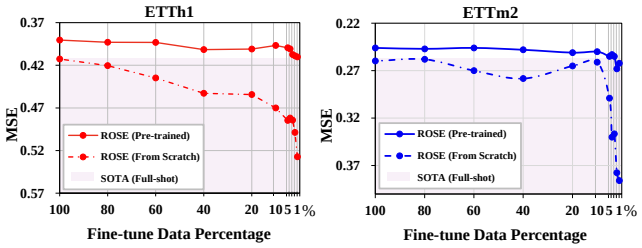


Figure 4: The forecasting results of ROSE obtained by training from scratch and fine-tuning from the pre-trained model. The right, upper corner is the best case.

4.2 ZERO-SHOT FORECASTING

Setting. In this section, to ensure a fair comparison, we conduct zero-shot predictions for each foundational model on downstream datasets not included in their pre-training data. It is worth noting that, unlike a few foundation models (Woo et al., 2024) that require much longer inputs to achieve better predictive performance, we fix the input length of all baselines to 512 without considering longer input lengths, as many real-world scenarios could offer very limited samples.

Results. As shown in Table 2, ROSE significantly outperforms across the majority of datasets, ***achieving an average reduction of 15% in Mean Squared Error (MSE).*** In comparison to Timer and Moirai, ROSE achieves average MSE reductions of 9% and 6%, respectively, and demonstrates a remarkable 43% relative improvement over Moment. Notably, ROSE stands out not only for its superior performance but also for its exceptional lightweight and efficient design, which sets it apart from other foundational models. Detailed analysis of these aspects will be presented in Section 4.3.

Table 2: The results for ROSE and other foundation models in zero-shot setting. The average results of all predicted lengths are listed here. We use '-' to indicate that the dataset has been involved in the model's pre-training, and thus not used for testing.

Models	ROSE		Timer		MOIRAI		Chronos		TimesFM		Moment	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.401**	**0.425**	0.451	0.463	0.475	0.443	0.560	0.452	0.489	0.444	0.708	0.580
ETTh2	**0.346**	**0.394**	0.366	0.408	0.379	0.396	0.392	0.397	0.396	0.405	0.392	0.430
ETTm1	0.525	0.471	0.544	0.476	0.714	0.507	0.636	0.495	**0.434**	**0.419**	0.697	0.555
ETTm2	**0.299**	**0.352**	0.360	0.386	0.343	0.356	0.313	0.363	0.320	0.353	0.319	0.360
Weather	**0.265**	0.305	0.292	0.312	0.267	**0.300**	0.288	0.310	-	-	0.291	0.323
Electricity	**0.234**	0.320	0.297	0.375	0.241	0.328	0.245	**0.312**	-	-	0.861	0.766
Traffic	**0.588**	0.412	0.613	**0.407**	-	-	0.371	0.370	-	-	1.411	0.804

4.3 Model Analysis

Efficiency analysis. To exhibit the performance and efficiency advantages of ROSE, we compare its parameter count to other foundation models and evaluate their performance and testing time averaged on ETTh1 and ETTh2 datasets in zero-setting. Similarly, for each specific model, we evaluate its parameter count as well as its performance in full-shot setting and training-to-testing time averaged on the same datasets. Specific implementation details and results can be found in Appendix A.2.

As shown in Figure 5, ROSE is a lightweight general model with 7.4M parameters and short inference time, which are only about one-tenth of the second fastest/smallest foundation model (Timer). Importantly, ROSE uses the least number of parameters among foundation models, with its parameter count approaching that of specific models, while exhibiting superior zero-shot performance. This is attributed to our proposed decomposed frequency learning that enhances the model's comprehension of time series. Concurrently, the TS-Register achieves the adaptive transfer thus efficiently adapting to downstream tasks without the need of scaling up to achieve strong generalizability. Compared to foundation models with large scale, ROSE may better meet the need for general models in real scenarios that require high computational and parameter efficiency as well as high prediction accuracy with scarce downstream data.

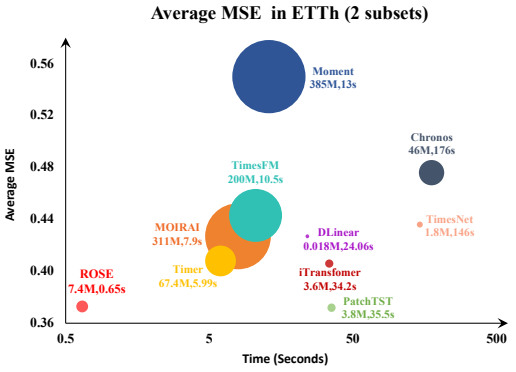


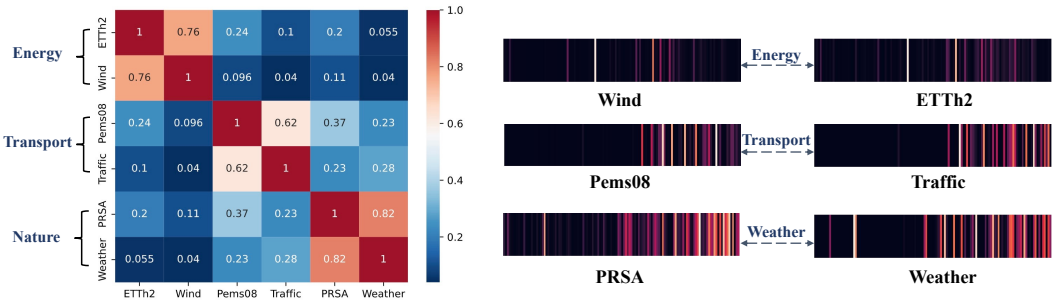
Figure 5: Model performance, number of parameter and efficiency comparison.

Visualization of TS-Register. To validate the TS-Register's capability to transfer domain-specific information adaptively from pre-training datasets to target datasets, we visualize the cosine similarity of register vector selections from datasets across different domains. As shown in Figure 6(a), the cosine similarity is higher for datasets within the same domain and lower between different domains. We also visualize the register vector selections from different datasets in Figures 6(b) and (c), where datasets from the same domain show similar visualizations. This confirms the TS-Register's capability of adaptive transfer from multi-source to target datasets across various domains.

Scalability and Sensitivity. The scalability analysis of ROSE's model size and pre-training data size are presented in Appendix A.4. The sensitivity analyses for the upper bound a of the thresholds, the number of masked series K_{f}, the number of register tokens N_{r}, the size of register H and number of selections k in Top-K strategy are presented in Appendix A.5.

4.4 Ablation Studies

Model architecture. To validate effectiveness of our model design, we perform ablation studies on TS-Register, prediction tasks, and reconstruction task in 10% few-shot setting. Table 3 shows the impact of each module. The TS-Register leverages multi-domain information during pre-training, aiding adaptive transfer to downstream datasets, as further discussed in Section 4.3. The prediction



(a) The cosine similarity of register vectors selection from various domains

(b) The register vectors selection from pretrain data

(c) The register vectors selection from target data

Figure 6: Visualization of TS-Register. The calculation of cosine similarity is in the Appendix A.3.

tasks enhance performance in data-scarce situations. Without it, performance significantly drops on ETTh1 and ETTh2 with limited samples. Without the reconstruction task, our model shows negative transfer effects on ETTm1 and ETTm2, likely due to the prediction task making the model more susceptible to pre-training data biases.

Masking method. To further validate the effectiveness of decomposed frequency learning, we replace the multi-frequency masking with different masking methods, including two mainstream time-domain methods: patch masking (Nie et al., 2022) and multi-patch masking (Dong et al., 2024), as well as random frequency masking (Chen et al., 2023b). The results in Table 4 show that random frequency masking and patch masking led to negative transfer on ETTm1 and ETTm2, likely due to significant disruption of the original time series, causing overfitting. In contrast, multi-patch masking and multi-frequency masking resulted in positive transfer across all datasets by preventing excessive disruption. Multi-frequency masking achieved better results, demonstrating its ability to help the model understand temporal patterns from a multi-frequency perspective. We also compare with some other pre-training tasks in Table 14 in Appendix A.10.3.

Table 3: Ablations on key components of model architecture, including TS-register, prediction task and reconstruction task. The average results of all predicted lengths are listed here.

Design		ETTm1		ETTm2		ETTh1		ETTh2	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ROSE		**0.349**	**0.372**	**0.250**	**0.308**	**0.397**	**0.419**	**0.335**	**0.380**
w/o	TS-Register	0.354	0.378	0.256	0.312	0.418	0.427	0.355	0.390
w/o	Prediction Task	0.360	0.384	0.257	0.314	0.422	0.438	0.372	0.410
w/o	Reconstruction Task	0.387	0.403	0.269	0.327	0.412	0.428	0.361	0.399
From scratch		0.371	0.391	0.261	0.318	0.470	0.480	0.400	0.425

Table 4: Ablations on decomposed frequency learning, where we replace Multi-freq masking with other masking methods. The average results of all predicted lengths are listed here.

Design		ETTm1		ETTm2		ETTh1		ETTh2	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ROSE		**0.349**	**0.372**	**0.250**	**0.308**	**0.397**	**0.419**	**0.335**	**0.380**
Replace Multi-Frequency Masking	Random Freq Masking	0.381	0.397	0.261	0.324	0.410	0.427	0.374	0.405
Replace Multi-Frequency Masking	Multi-Patch Masking	0.356	0.379	0.259	0.316	0.404	0.426	0.349	0.389
Replace Multi-Frequency Masking	Patch Masking	0.378	0.400	0.261	0.319	0.408	0.432	0.375	0.407
From scratch		0.371	0.391	0.261	0.318	0.470	0.480	0.400	0.425

5 Conclusion and Future Work

In this work, we propose ROSE, a novel general model, addressing the challenges of leveraging multi-domain datasets for enhancing downstream prediction task performance. ROSE utilizes decomposed frequency learning and TS-Register to capture generalized and domain-specific representations, enabling improved fine-tuning results, especially in data-scarce scenarios. Our experiments demonstrate ROSE's superior performance over baselines with both full-data and few-data fine-tuning, as well as its impressive zero-shot capabilities. Future efforts will concentrate on expanding pre-training datasets and extending ROSE's applicability across diverse time series analysis tasks, e.g., classification. We provide our code at `https://anonymous.4open.science/r/ROSE-A235`.

6 REPRODUCIBILITY

Our work meets reproducibility requirements. Specifically, you can download our evaluation datasets in a standardized format from the public link (Wang et al., 2024a): `https://drive.google.com/drive/folders/13Cg1KYOlzM5C7K8gK8NfC-F3EYxkM3D2`, and we provide our code at `https://anonymous.4open.science/r/ROSE-A235`.

REFERENCES

Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S Jensen, Zhenli Sheng, et al. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *arXiv preprint arXiv:2403.20150*, 2024.

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

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

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. *Advances in Neural Information Processing Systems*, 35:3988–4003, 2022.

Xingyu Zhang, Siyu Zhao, Zeen Song, Huijie Guo, Jianqi Zhang, Changwen Zheng, and Wenwen Qiang. Not all frequencies are created equal: Towards a dynamic fusion of frequencies in time-series forecasting. In *ACM Multimedia 2024*, 2024.

Jiaxiang Dong, Haixu Wu, Haoran Zhang, Li Zhang, Jianmin Wang, and Mingsheng Long. Simmtm: A simple pre-training framework for masked time-series modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

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

Seunghan Lee, Taeyoung Park, and Kibok Lee. Learning to embed time series patches independently. *arXiv preprint arXiv:2312.16427*, 2023.

Liyue Chen, Linian Wang, Jinyu Xu, Shuai Chen, Weiqiang Wang, Wenbiao Zhao, Qiyu Li, and Leye Wang. Knowledge-inspired subdomain adaptation for cross-domain knowledge transfer. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 234–244, 2023a.

Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36, 2024.

George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968.

Razvan-Gabriel Cirstea, Bin Yang, and Chenjuan Guo. Graph attention recurrent neural networks for correlated time series forecasting. In *MileTS19@KDD*, 2019.

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3): 1181–1191, 2020.

Donghao Luo and Xue Wang. Moderntcn: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024.

Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022a.

Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2022.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The eleventh international conference on learning representations*, 2022.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

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

Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. 2024.

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

Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Vincent Hassen, Anderson Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

Azul Garza and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Harshavardhan Kamarthi and B Aditya Prakash. Large pre-trained time series models for cross-domain time series analysis tasks. *arXiv preprint arXiv:2311.11413*, 2023.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023a.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

E Oran Brigham and RE Morrow. The fast fourier transform. *IEEE spectrum*, 4(12):63–70, 1967.

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. $\mathbf{S}^2$ ip-llm: Semantic space informed prompt learning with llm for time series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023b.

Muxi Chen, Zhijian Xu, Ailing Zeng, and Qiang Xu. Fraug: Frequency domain augmentation for time series forecasting. *arXiv preprint arXiv:2302.09292*, 2023b.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024a.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I Webb, Rob J Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*, 2021.

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.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

Shuyi Zhang, Bin Guo, Anlan Dong, Jing He, Ziping Xu, and Song Xi Chen. Cautionary tales on air-quality improvement in beijing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2205):20170457, 2017.

Yihe Wang, Yu Han, Haishuai Wang, and Xiang Zhang. Contrast everything: A hierarchical contrastive framework for medical time-series. *Advances in Neural Information Processing Systems*, 36, 2024b.

Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022b.

Michael W McCracken and Serena Ng. Fred-md: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4):574–589, 2016.

Souhaib Ben Taieb, Gianluca Bontempi, Amir F Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert systems with applications*, 39(8):7067–7083, 2012.

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

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

Spyros Makridakis. M4 dataset, 2018. https://github.com/M4Competition/M4-methods/tree/master/Dataset,.

Xiyuan Zhang, Ranak Roy Chowdhury, Jingbo Shang, Rajesh Gupta, and Dezhi Hong. Towards diverse and coherent augmentation for time-series forecasting. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8980–8987, 2022.

A APPENDIX

A.1 IMPLEMENTATION DETAILS

A.1.1 PRE-TRAINING DATASETS

We use multi-source datasets in pre-training which contain subsets of Monash (Godahewa et al., 2021), UEA (Bagnall et al., 2018) and UCR (Dau et al., 2019) time series datasets, as well as some other time series classical datasets (Zhang et al., 2017; Wang et al., 2024b; Liu et al., 2022b; McCracken and Ng, 2016; Taieb et al., 2012). The final list of all pre-training datasets is shown in Table 6. There is no overlap between the pre-training datasets and the target datasets. It is worth noting that the dataset weather in the pre-training dataset is a univariate dataset, which is different to the multivariate dataset weather in the target task. The pre-trained datasets can be categorized into 6 different domains according to their sources: Energy, Nature, Health, Transport, and Web. The sampling frequencies of the datasets show a remarkable diversity, ranging from millisecond samples to monthly samples, which reflects the diverse application scenarios and complexity of the real world. For all pre-training datasets, we split them into univariate sequences and train them in a channel-independent manner.

A.1.2 EVALUATION DATASETS

We use the following 7 multivariate time-series datasets for downstream fine-tuning and forecasting: ETT datasets[1] contain 7 variates collected from two different electric transformers from July 2016 to July 2018. It consists of four subsets, of which ETTh1/ETTh2 are recorded hourly and ETTm1/ETTm2 are recorded every 15 minutes. Traffic[2] contains road occupancy rates measured by 862 sensors on freeways in the San Francisco Bay Area from 2015 to 2016, recorded hourly. Weather[3] collects 21 meteorological indicators, such as temperature and barometric pressure, for Germany in 2020, recorded every 10 minutes. Electricity[4] contains the electricity consumption of 321 customers from July 2016 to July 2019, recorded hourly. We split each evaluation dataset into train-validation-test sets and detailed statistics of evaluation datasets are shown in Table 5.

Table 5: The statistics of evaluation datasets.

Dataset	ETTm1	ETTm2	ETTh1	ETTh2	Traffic	Weather	Electricity
Variables	7	7	7	7	862	21	321
Timestamps	69680	69680	17420	17420	17544	52696	26304
Split Ratio	6:2:2	6:2:2	6:2:2	6:2:2	7:1:2	7:1:2	7:1:2

A.1.3 SETTING

We implemented ROSE in PyTorch (Paszke et al., 2019) and all the experiments were conducted on 8 NVIDIA A800 80GB GPU. We used ADAM (Kingma and Ba, 2014) with an initial learning rate of 5×10^{-4} and implemented learning rate decay using the StepLR method to implement learning rate decaying pre-training. By default, ROSE contains 3 encoder layers and 3 decoder layers with head number of 16 and the dimension of latent space $D = 256$. The patch size for patching is set to 64.

Pre-training. We use $N_{\mathrm{r}} = 3$ as the number of register tokens and $P = 8$ as the path tokens. We set the input length to 512 for the supervised prediction task with target lengths of 96, 192, 336, and 720. We also set the input length to 512 and mask number $K_{\mathbf{f}} = 4$. The batch size is set to 8192 in pre-training.

Fine-tuning. We fix the lookback window to 512, and perform predictions with target lengths of 96, 192, 336, and 720, respectively. The number of register tokens N_{r} and patch tokens P is the same as in pre-training, and the parameter $k = 3$ in TopK is set when selection vectors are performed in the register.

[1] https://github.com/zhouhaoyi/ETDataset

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

[3] https://www.bgc-jena.mpg.de/wetter/

[4] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

Table 6: List of pretraining datasets.

Domain	Dataset	Frequency	Time Pionts	Source
Energy	Aus. Electricity Demand	Half Hourly	1155264	Monash(Godahewa et al., 2021)
	Wind	4 Seconds	7397147	Monash(Godahewa et al., 2021)
	Wind Farms	Minutely	172178060	Monash(Godahewa et al., 2021)
	Solar	10 Minutes	7200720	Monash(Godahewa et al., 2021)
	Solar Power	4 Seconds	7397222	Monash(Godahewa et al., 2021)
	London Smart Meters	Half Hourly	166527216	Monash(Godahewa et al., 2021)
Nature	Phoneme	-	2160640	UCRDau et al. (2019)
	EigenWorms	-	27947136	UEA(Bagnall et al., 2018)
	PRSA	Hourly	4628448	(Zhang et al., 2017)
	Temperature Rain	Daily	23252200	Monash(Godahewa et al., 2021)
	StarLightCurves	-	9457664	UCR(Dau et al., 2019)
	Worms	0.033 Seconds	232200	UCR(Dau et al., 2019)
	Saugeen River Flow	Daily	23741	Monash(Godahewa et al., 2021)
	Sunspot	Daily	73924	Monash(Godahewa et al., 2021)
	Weather	Daily	43032000	Monash(Godahewa et al., 2021)
	KDD Cup 2018	Daily	2942364	MonashGodahewa et al. (2021)
	US Births	Daily	7305	Monash(Godahewa et al., 2021)
Health	MotorImagery	0.001 Seconds	72576000	UEA(Bagnall et al., 2018)
	SelfRegulationSCP1	0.004 Seconds	3015936	UEA(Bagnall et al., 2018)
	SelfRegulationSCP2	0.004 Seconds	3064320	UEA(Bagnall et al., 2018)
	AtrialFibrillation	0.008 Seconds	38400	UEA(Bagnall et al., 2018)
	PigArtPressure	-	624000	UCR(Dau et al., 2019)
	PIGCVP	-	624000	UCR(Dau et al., 2019)
	TDbrain	0.002 Seconds	79232703	(Wang et al., 2024b)
Transport	Pems03	5 Minute	9382464	(Liu et al., 2022b)
	Pems04	5 Minute	5216544	(Liu et al., 2022b)
	Pems07	5 Minute	24921792	(Liu et al., 2022b)
	Pems08	5 Minute	3035520	(Liu et al., 2022b)
	Pems-bay	5 Minute	16937700	(Liu et al., 2022b)
	Pedestrian_Counts	Hourly	3132346	Monash(Godahewa et al., 2021)
Web	Web Traffic	Daily	116485589	Monash(Godahewa et al., 2021)
Economic	FRED_MD	Monthly	77896	(McCracken and Ng, 2016)
	Bitcoin	Daily	75364	Monash(Godahewa et al., 2021)
	NN5	Daily	87801	(Taieb et al., 2012)

A.1.4 BASELINES

We select the state-of-the-art models as our baselines in full-shot and few-shot setting, including four specific models: iTransformer (Liu et al., 2023), PatchTST (Nie et al., 2022), TimesNet (Wu et al., 2022), and DLinear (Zeng et al., 2023), and two LLM-based models: GPT4TS (Zhou et al., 2024) and $\mathbf{S}^2$IP-LLM (Pan et al., 2024). In addition, we selected five foundation models for comparison in zero-shot setting, including Timer (Liu et al., 2024), MOIRAI (Woo et al., 2024), Chronos (Ansari et al., 2024), TimesFM (Das et al., 2023b) and Moment (Goswami et al., 2024). The specific code base for these models is listed in Table 7:

A.2 EFFICIENCY ANALYSIS

As an important aspect of foundation models, inference efficiency is crucial. Therefore, we evaluate the testing time of ROSE and five foundation models in the ETTh1 and ETTh2 dataset in zero-shot setting. Similarly, we evaluate the time of the entire process of training, validation, and testing for four specific models in the same datasets in full-shot setting. The above experiments all set the batch size to 32. The specific results are shown in Table 8 and Figure 5. We observe that ROSE maintains its advantage in zero-shot performance while also being significantly faster compared to the baselines,

Table 7: Code repositories for baselines.

Model Types	Models	Code Repositories
Small Model	iTransformer	https://github.com/thuml/iTransformer
	PatchTST	https://github.com/yuqinie98/PatchTST
	TimesNet	https://github.com/thuml/TimesNet
	Dlinear	https://github.com/cure-lab/LTSF-Linear
Foundation Model	Timer	https://github.com/thuml/Large-Time-Series-Model
	MOIRAI	https://github.com/redoules/moirai
	Chronos	https://github.com/amazon-science/chronos-forecasting
	TimesFM	https://github.com/google-research/timesfm/
	Moment	https://anonymous.4open.science/r/BETT-773F/README.md
LLM-based Model	GPT4TS	https://github.com/DAMO-DI-ML/NeurIPS2023-One-Fits-All
	S2IP-LLM	https://github.com/panzijie825/S2IP-LLM

even being approximately ten times faster than the second-fastest foundation model, Timer. This raises our reflection on whether time-series foundation models require extremely large parameter sizes and whether existing time-series foundation models have validated their architectures' scaling laws on time-series data.

Table 8: Efficiency analysis.

Model	Parameters	Pre-train datasize	Averaged time
ROSE	7.4M	0.89B	0.652s
MOIRAI	311M	27B	7.920s
Timer	67.4M	1B	5.989s
Chronos	46M	84B	176s
TimesFM	200M	100B	10.5s
Moment	385M	1.13B	13s
Itransformer	3.8M	-	34.18s
PatchTST	3.2M	-	35.47s
TimesNet	1.8M	-	146s
Dlinear	0.018M	-	24.06s

A.3 Calculation of cosine similarity

In Figure 6, we visualize the cosine similarity of register vector selections. For each sample in a dataset, during the inference process, k vectors are selected from the register based on the Top-K strategy. We iterate through all samples in the dataset and count the number of times each vector is selected, which allows us to obtain a record vector of length equivalent to the size of the register for each dataset. The i_{th} position in the record vector represents the number of times the i_{th} vector in the register has been selected by samples in the dataset. For a pair of datasets, we can obtain a unique record vector for each dataset, and then we are able to calculate the cosine similarity of two vectors.

A.4 Scalability

Scalability is crucial for a general model, enabling significant performance improvements by expanding pre-training data and model sizes. To investigate the scalability of ROSE, we increased both the model size and dataset size and evaluated its predictive performance on four ETT datasets.

Model size. Constrained by computational resources, we use 40% pre-training datasets. The results are shown in Figure 7(a) and (b). When maintaining the model dimension, we increased the model layers, increasing model parameters from 2M to 4.5M. This led to 10.37% and 9.34% improvements in the few-shot scenario with 5% and 10% downstream data, respectively.

Data size. When keeping the model size, we increase the size of the pre-training datasets from 178M to 887M. The results are shown in Figure 7(c) and (d). The performance of our model steadily improves with the increase in dataset size and achieves improvements of 7.4% and 4.8% respectively.

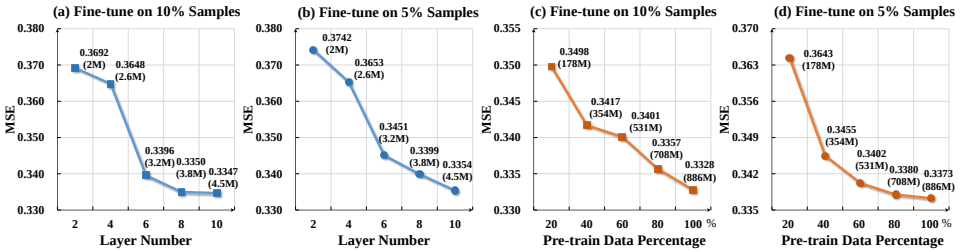


Figure 7: (a)/(b): Larger ROSE demonstrates better performance on downstream forecasting. (c)/(d): ROSE pre-trained on larger datasets demonstrates better performance on downstream forecasting.

A.5 SENSITIVITY

We perform the sensitivity analyses for the upper bound a of the thresholds, the number of masked series K_{f}, the number of register tokens N_{r}, the size of register H and the number of selections k in Top-K strategy. All the sensitivity experiments present the average results on the four ETT datasets: ETTh1, ETTh2, ETTm1 and ETTm2 under 10% few-shot setting.

Number of masked series. As described in Section 3.2, we propose decomposed frequency learning, which employs multiple thresholds to randomly mask high and low frequencies in the frequency domain, thereby decomposing the original time series into multiple frequency components. This allows the model to understand the time series from multiple frequency perspectives. In this experiment, we study the influence of the number of masked series K_{f} on downstream performance. We train ROSE with 1, 2, 3, 4, 5, or 6 mask series. We report the results of this analysis in Figure 8(a). We find that as the number of masked sequences increases, the downstream performance gradually improves. This is because the model can better understand the time series from the decomposed frequency components, which enhances the model's generalization ability. However, more masked series do not bring better downstream performance. This could be due to an excessive number of masked sequences leading to information redundancy. In all our experiments, we keep 4 mask series.

Number of register tokens. The TS-register module presented in Section 3.3 supports the configuration of an arbitrary number of register tokens. In Figure 8(b), we visualize the relationship between the performance on the ETT datasets under a 10% few-shot setting and the number of register tokens. It is observed that when the number of register tokens ranges from 1 to 6, the model's performance remains relatively stable, with an optimal outcome achieved when the number is set to 3. This phenomenon may be because when the number of register tokens is too small, they contain insufficient domain-specific information, which limits their effectiveness in enhancing the model's performance. Conversely, an excess of register tokens may introduce redundant information, hindering the accurate representation of domain-specific information. Additionally, we compared the results without the adjustment of a low-rank matrix on the register tokens and found that the incorporation of a low-rank matrix adjustment led to improvements across all quantities of register tokens. This finding underscores the significance of utilizing a low-rank matrix to supplement the register tokens with downstream data-specific information.

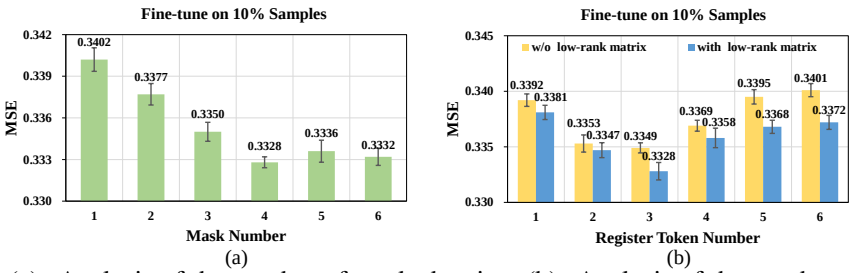


Figure 8: (a): Analysis of the number of masked series. (b): Analysis of the number of register tokens.

Thresholds upper bound. Figure 9(a) illustrates the relationship between threshold upper bound and model performance. We have observed that the upper bound of the threshold has a minimal impact on the model's performance. Generally, the information density is higher in low-frequency components compared to high-frequency ones. Therefore, the upper bound of the threshold should be biased

towards the low-frequency range to balance the information content between low-frequency and high-frequency components. However, this bias should not be excessive. Our experiments indicate that an upper bound of $L/10$ performs worse than $L/5$ as an overly left-skewed threshold results in insufficient information in the low-frequency range, making the reconstruction task either too difficult or too simple. Based on our findings, we recommend using $L/5$ as the upper bound for the threshold.

Register size. Figure 9(b) illustrates the relationship between register size and model performance. The register size determines the upper limit of domain-specific information that the register can store. We can observe that there is a significant improvement in the model effect when the register size is increased from 32 to 128. When the register size exceeds 128, the improvement of the model effect with the increase of register size is no longer obvious. Therefore, we believe that 128 is an appropriate register size for the current pre-training datasets.

Number of selections in Top-K strategy. Figure 9(c) illustrates the relationship between the number of selections k in Top-K strategy and model performance when we use the register to realize adaptive transfer of domain-specific information in downstream tasks. It can be seen that the model effect performance peaks at 3 tokens at k = 3, which has some advantages over selecting once (k = 1), indicating that the TopK strategy can compensate for the problem of incomplete matching of upstream and downstream domains to some extent. However, too large k will also introduce redundant information and limit the accuracy of domain-specific information transfer.

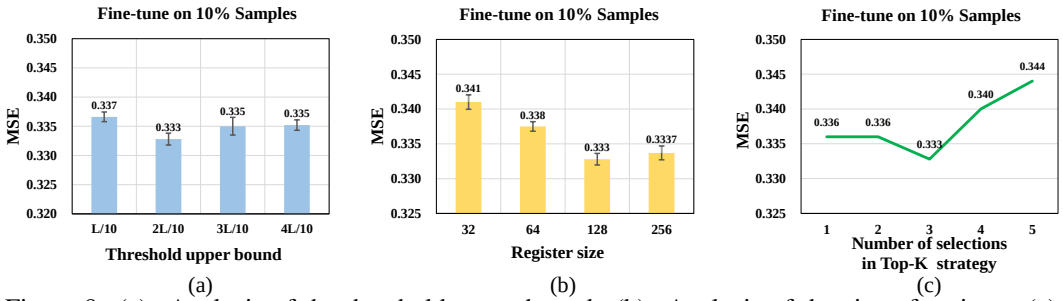


Figure 9: (a): Analysis of the threshold upper bound. (b): Analysis of the size of register. (c): Analysis of the number of selections in Top-K strategy.

A.6 SHORT-TERM FORECASTING

We also try to apply ROSE to short-term forecasting on the M4 (Makridakis, 2018) dataset, which contains the yearly, quarterly and monthly collected univariate marketing data. We follow TimesNet's (Wu et al., 2022) setting and metrics (SMAPE, MASE and OWA) for testing. As shown in Table 9, ROSE also exhibits competitive performance on the M4 dataset compared to the baselines.

Table 9: Full results on Short-term forecasting.

Metric	ROSE			iTransformer			PatchTST			Timesnet			Dlinear			GPT4TS			S2IP-LLM		
	SMAPE	MASE	OWA	SMAPE	MASE	OWA	SMAPE	MASE	OWA	SMAPE	MASE	OWA	SMAPE	MASE	OWA	SMAPE	MASE	OWA	SMAPE	MASE	OWA
Yearly	13.302	3.014	0.833	**13.238**	**2.952**	0.823	16.766	4.331	1.018	13.387	2.996	**0.786**	16.965	4.283	1.058	13.531	3.015	0.793	13.413	3.024	0.792
Quarterly	**9.998**	**1.165**	**0.885**	10.001	1.278	0.949	12.132	1.513	0.966	10.100	1.182	0.890	12.145	1.520	1.106	10.100	1.194	0.898	10.352	1.228	0.922
Monthly	**12.650**	**0.915**	**0.866**	13.399	1.031	0.949	13.428	0.997	0.948	12.670	0.933	0.933	13.514	1.037	0.956	12.894	0.956	0.897	12.995	0.970	0.910
Others	**4.668**	**3.126**	**1.020**	6.558	4.511	1.401	6.667	4.834	1.417	4.891	3.302	1.035	6.709	4.953	1.487	4.940	3.228	1.029	4.805	3.247	1.071

A.7 MODEL GENERALITY

We evaluate the effectiveness of our proposed multi-frequency masking on Transformer-based models and CNN-based models, whose results are shown in Table 10. It is notable that multi-frequency masking consistently improves these forecasting models. Specifically, it achieves average improvements of 6.3%, 3.7%, 1.5% in Autoformer (Wu et al., 2021), TimesNet (Wu et al., 2022), and PatchTST (Nie et al., 2022), respectively. This indicates that multi-frequency Masking can be widely utilized across various time series forecasting models to learn generalized time series representations and improve prediction accuracy.

Table 10: Performance of multi-frequency masking.

Datasets	ETTm1		ETTm2		ETTh1		ETTh2	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Autoformer	0.600	0.521	0.328	0.365	0.493	0.487	0.452	0.458
+Multi-frequency Masking	**0.549**	**0.488**	**0.306**	**0.349**	**0.474**	**0.478**	**0.406**	**0.425**
TimesNet	0.400	0.406	0.291	0.333	0.458	0.450	0.414	0.427
+Multi-frequency Masking	**0.386**	**0.398**	**0.282**	**0.324**	**0.446**	**0.438**	**0.386**	**0.403**
PatchTST	0.353	0.382	0.256	0.317	0.413	0.434	**0.331**	0.381
+Multi-frequency Masking	**0.347**	**0.372**	**0.252**	**0.308**	**0.405**	**0.424**	0.337	**0.379**

A.8 RESULTS DEVIATION

We have conducted ROSE three times with different random seeds and have recorded the standard deviations for both the full-shot setting and the 10% few-shot setting, as illustrated in Table 11. As the baselines didn't report deviations in the original paper, we only reported the deviations of the PatchTST in the full-shot setting as a comparison. It can be observed that ROSE exhibits stable performance.

Table 11: Results deviation.

Models	ROSE		ROSE (10%)		PatchTST		confidence interval
Metric	MSE	MAE	MSE	MAE	MSE	MAE	-
ETTm1	0.342±0.003	0.367±0.002	0.349±0.003	0.372±0.002	0.349± 0.004	0.383±0.003	99%
ETTm2	0.246±0.002	0.303±0.004	0.249±0.002	0.308±0.002	0.255±0.002	0.314±0.003	99%
ETTh1	0.392±0.004	0.413±0.004	0.397±0.003	0.419±0.003	0.411±0.003	0.432±0.005	99%
ETTh2	0.330±0.003	0.374±0.002	0.335±0.004	0.380±0.003	0.348±0.004	0.390±0.004	99%
Traffic	0.391±0.008	0.266±0.005	0.418±0.011	0.278±0.006	0.404±0.009	0.283±0.002	99%
Weather	0.217±0.008	0.250±0.007	0.224±0.007	0.252±0.009	0.223±0.011	0.263±0.014	99%
Electricity	0.156±0.007	0.249±0.009	0.164±0.004	0.253±0.004	0.163±0.009	0.261±0.013	99%

A.9 VISUALIZATION

A.9.1 VISUALIZATION ANALYSIS

To showcase the benefits of cross-domain pre-training, we performed visualizations in both the zero-shot setting and full-shot setting.

Zero-shot: We pre-train the baselines iTransformer and PatchTST on the energy domain dataset ETTm1 and test their zero-shot performance on two different domains (weather, traffic) . ROSE, without fine-tuning, is evaluated to the same two test-sets. As shown in the Figure 10, we find that the baselines generally perform worse during domain shift due to their poor generalization. However, ROSE excels in scenarios across all domains, which demonstrates the benefits of cross-domain pre-training for improving generalization.

Full-shot: We train the baselines on the train-set of downstream dataset ETTh2 and fine-tune ROSE on the same train-set. As shown in the Figure 11, We find that the baselines is limited by data diversity, leading to poor performance on patterns which rarely appear. However, ROSE excels in these cases, as the cross-domain pre-training allows ROSE to learn diverse temporal patterns, and helps ROSE to predict the patterns which rarely appear in the downstream train-set well.

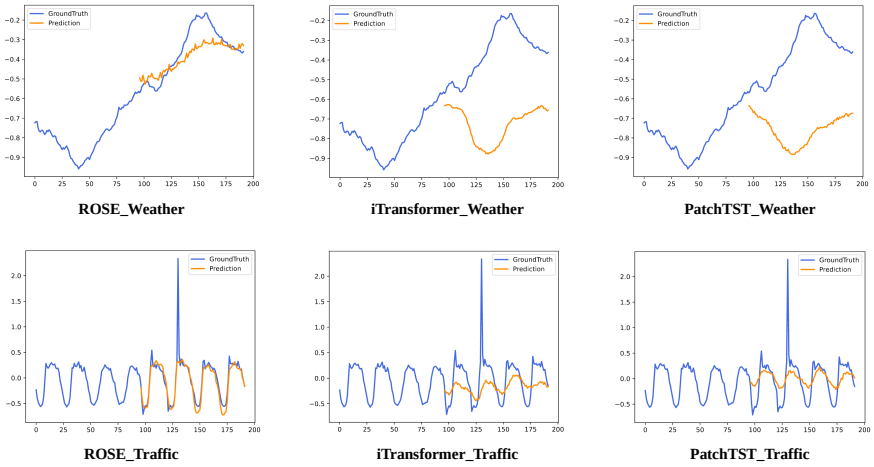


Figure 10: Visualization comparison of ROSE with cross-domain pre-training and other SOTA baselines in the zero-shot setting for three domain datasets.

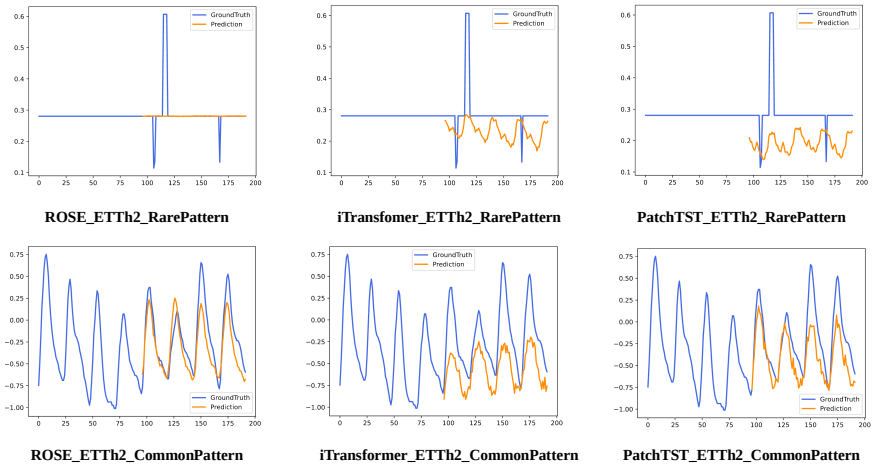


Figure 11: Visualization comparison of ROSE with cross-domain pre-training and other SOTA baselines in the full-shot setting for rare and common patterns.

A.9.2 VISUALIZATION SHOWCASE

To provide a distinct comparison among different models, we present visualizations of the forecasting results on the ETTh2 dataset and the weather dataset in different settings, as shown in Figures 12 to Figures 15, given by the following models: DLinear (Zeng et al., 2023), TimesNet (Wu et al., 2022), iTransfomrer (Liu et al., 2023), and PatchTST (Nie et al., 2022). Among the methods, ROSE demonstrates the most accurate prediction ability.

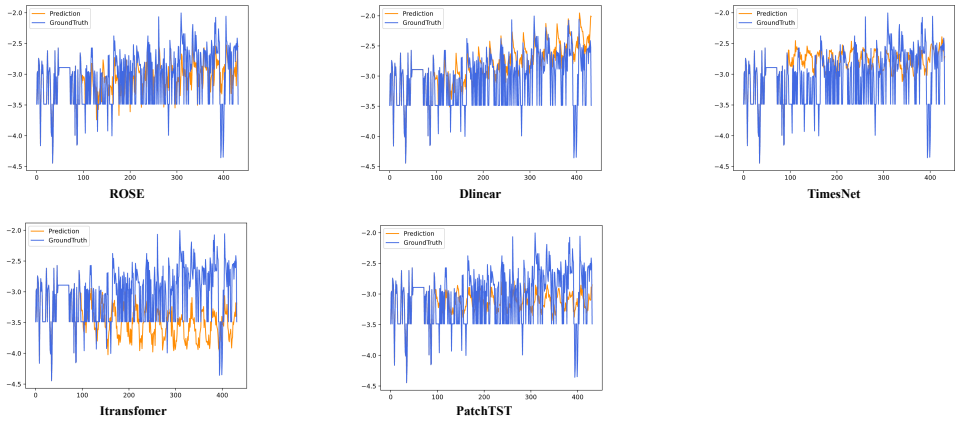


Figure 12: Visualization of input-512 and predict-336 forecasting results on the ETTh2 dataset in full-shot setting.

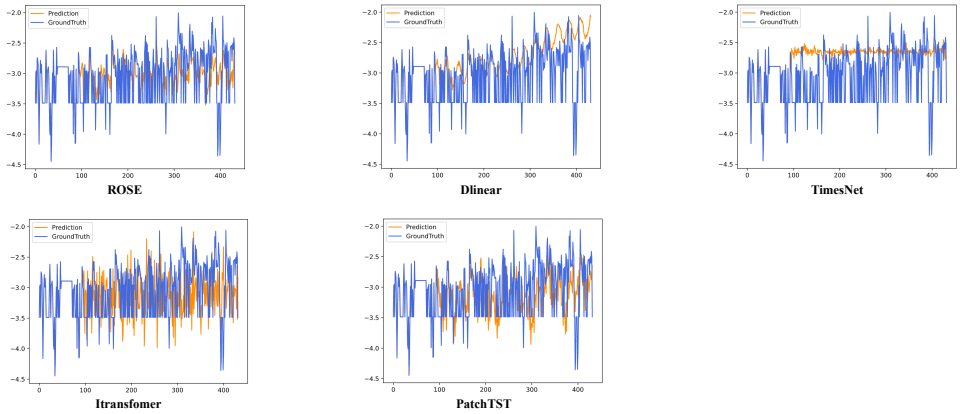


Figure 13: Visualization of input-512 and predict-336 forecasting results on the ETTh2 dataset in 10% few-shot setting.

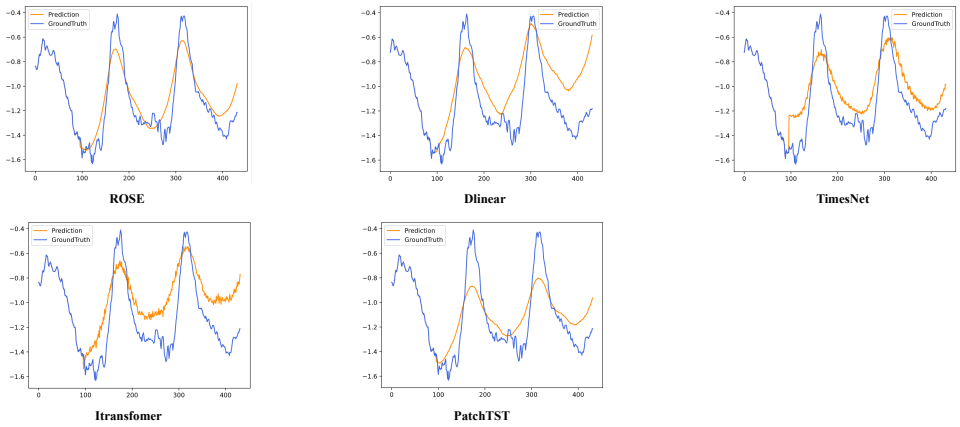


Figure 14: Visualization of input-512 and predict-336 forecasting results on the weather dataset in full-shot setting.

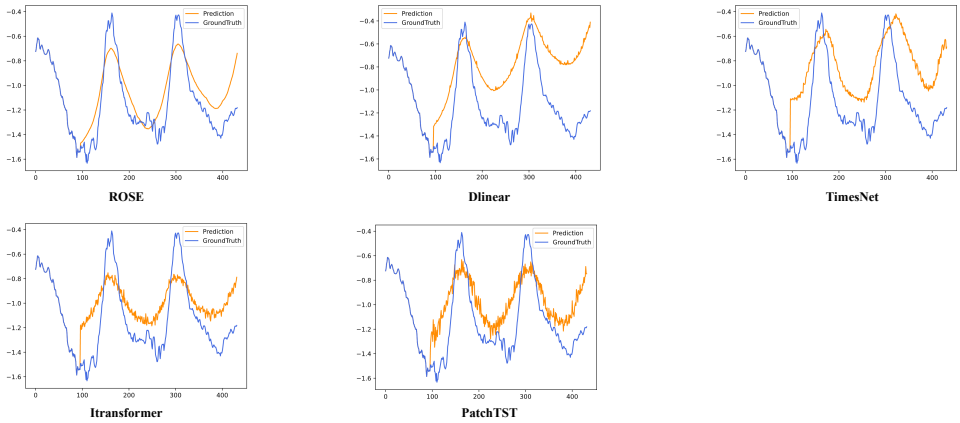


Figure 15: Visualization of input-512 and predict-336 forecasting results on the weather dataset in 10% few-shot setting.

A.10 FULL RESULTS

A.10.1 FULL-SHOT RESULTS

Table 12: Full results in full-shot setting.

Models		ROSE		ITransformer		PatchTST		Timesnet		Dlinear		GPT4TS		S^2IP-LLM	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.354**	**0.385**	0.386	0.405	0.370	0.400	0.470	0.470	0.367	0.396	0.376	0.397	0.366	0.396
	192	**0.389**	**0.407**	0.424	0.440	0.413	0.429	0.568	0.523	0.400	0.417	0.416	0.418	0.401	0.420
	336	**0.406**	**0.422**	0.449	0.460	0.422	0.440	0.595	0.547	0.428	0.439	0.442	0.433	0.412	0.431
	720	**0.413**	**0.443**	0.495	0.487	0.447	0.468	0.694	0.591	0.468	0.491	0.477	0.456	0.440	0.458
	avg	**0.391**	**0.414**	0.439	0.448	0.413	0.434	0.582	0.533	0.416	0.436	0.427	0.426	0.406	0.427
ETTh2	96	**0.265**	**0.320**	0.297	0.348	0.274	0.337	0.351	0.399	0.302	0.368	0.285	0.342	0.278	0.340
	192	0.328	**0.369**	0.371	0.403	0.341	0.382	0.394	0.429	0.404	0.433	0.354	0.389	**0.246**	0.385
	336	0.353	0.391	0.404	0.428	**0.329**	**0.384**	0.415	0.443	0.511	0.498	0.373	0.407	0.367	0.406
	720	**0.376**	**0.417**	0.424	0.444	0.379	0.422	0.477	0.481	0.815	0.640	0.406	0.441	0.400	0.436
	avg	**0.331**	**0.374**	0.374	0.406	**0.331**	0.381	0.409	0.438	0.508	0.485	0.354	0.394	0.347	0.391
ETTm1	96	**0.275**	0.328	0.300	0.353	0.293	0.346	0.405	0.421	0.303	0.346	0.292	**0.262**	0.288	0.346
	192	0.324	0.358	0.345	0.382	0.333	0.370	0.508	0.473	0.335	0.365	0.332	**0.301**	**0.323**	0.365
	336	**0.354**	0.377	0.374	0.398	0.369	0.392	0.523	0.479	0.365	0.384	0.366	**0.341**	0.359	0.390
	720	0.411	0.407	0.429	0.430	0.416	0.420	0.523	0.484	0.418	0.415	0.417	**0.401**	**0.403**	0.418
	avg	**0.341**	**0.367**	0.362	0.391	0.353	0.382	0.490	0.464	0.356	0.378	0.352	0.383	0.343	0.379
ETTm2	96	**0.157**	**0.243**	0.175	0.266	0.166	0.256	0.233	0.305	0.164	0.255	0.173	0.262	0.165	0.257
	192	**0.213**	**0.283**	0.242	0.312	0.223	0.296	0.265	0.328	0.224	0.304	0.229	0.301	0.222	0.299
	336	**0.266**	**0.319**	0.282	0.340	0.274	0.329	0.379	0.392	0.277	0.339	0.286	0.341	0.277	0.330
	720	**0.347**	**0.373**	0.378	0.398	0.362	0.385	0.390	0.407	0.371	0.401	0.378	0.401	0.363	0.390
	avg	**0.246**	**0.305**	0.269	0.329	0.256	0.317	0.317	0.358	0.259	0.325	0.266	0.326	0.257	0.319
Weather	96	**0.145**	**0.182**	0.159	0.208	0.149	0.198	0.193	0.244	0.170	0.230	0.162	0.212	**0.145**	0.195
	192	**0.183**	**0.226**	0.200	0.248	0.194	0.241	0.320	0.329	0.212	0.267	0.204	0.248	0.190	0.235
	336	**0.232**	**0.267**	0.253	0.289	0.245	0.282	0.363	0.366	0.257	0.305	0.254	0.286	0.243	0.280
	720	**0.309**	0.327	0.321	0.338	0.314	0.334	0.440	0.404	0.318	0.356	0.326	0.337	0.312	**0.326**
	avg	**0.217**	**0.251**	0.233	0.271	0.226	0.264	0.329	0.336	0.239	0.289	0.237	0.270	0.222	0.259
Electricity	96	**0.125**	**0.220**	0.138	0.237	0.129	0.222	0.182	0.287	0.141	0.241	0.139	0.238	0.135	0.230
	192	**0.142**	**0.235**	0.157	0.256	0.147	0.240	0.193	0.293	0.154	0.254	0.153	0.251	0.149	0.247
	336	**0.162**	**0.252**	0.167	0.264	0.163	0.259	0.196	0.298	0.168	0.271	0.169	0.266	0.167	0.266
	720	**0.191**	**0.284**	0.194	0.286	0.197	0.290	0.209	0.307	0.203	0.303	0.206	0.297	0.200	0.287
	avg	**0.155**	**0.248**	0.164	0.261	0.159	0.253	0.195	0.296	0.166	0.267	0.167	0.263	0.161	0.257
Traffic	96	**0.354**	0.252	0.363	0.265	0.360	**0.249**	0.611	0.323	0.411	0.294	0.388	0.282	0.379	0.274
	192	**0.377**	0.257	0.385	0.273	0.379	**0.256**	0.609	0.327	0.421	0.298	0.407	0.290	0.397	0.282
	336	0.396	**0.262**	0.396	0.277	**0.392**	0.264	0.616	0.335	0.431	0.304	0.412	0.294	0.407	0.289
	720	0.434	**0.283**	0.445	0.312	**0.432**	0.286	0.656	0.349	0.468	0.325	0.450	0.312	0.440	0.301
	avg	**0.390**	**0.264**	0.397	0.282	0.391	**0.264**	0.623	0.333	0.433	0.305	0.414	0.294	0.405	0.286

A.10.2 FEW-SHOT RESULTS

Table 13: Full results in 10% few-shot setting

Models		ROSE		ITransformer		PatchTST		Timesnet		Dlinear		GPT4TS		S^2IP-LLM	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.367**	**0.395**	0.442	0.464	0.458	0.463	0.579	0.522	1.355	0.816	0.458	0.456	0.481	0.474
	192	**0.399**	**0.416**	0.476	0.475	0.481	0.490	0.641	0.553	1.210	0.825	0.570	0.516	0.518	0.491
	336	**0.405**	**0.423**	0.486	0.482	0.465	0.475	0.721	0.582	1.487	0.914	0.608	0.535	0.664	0.570
	720	**0.416**	**0.443**	0.509	0.506	0.478	0.492	0.630	0.574	1.369	0.826	0.725	0.591	0.711	0.584
	avg	**0.397**	**0.419**	0.478	0.482	0.470	0.480	0.643	0.558	1.355	0.845	0.590	0.525	0.593	0.529
ETTh2	96	**0.273**	**0.332**	0.333	0.385	0.350	0.389	0.378	0.413	1.628	0.724	0.331	0.374	0.354	0.400
	192	**0.334**	**0.376**	0.402	0.428	0.416	0.426	0.463	0.460	1.388	0.713	0.402	0.411	0.400	0.423
	336	**0.358**	**0.397**	0.438	0.452	0.401	0.429	0.507	0.495	1.595	0.772	0.406	0.433	0.442	0.450
	720	**0.376**	**0.417**	0.466	0.477	0.436	0.457	0.516	0.501	1.664	0.857	0.449	0.464	0.480	0.486
	avg	**0.335**	**0.380**	0.410	0.436	0.401	0.425	0.466	0.467	1.569	0.766	0.397	0.421	0.419	0.439
ETTm1	96	**0.287**	**0.336**	0.353	0.392	0.317	0.363	0.481	0.446	0.454	0.475	0.390	0.404	0.388	0.401
	192	**0.331**	**0.362**	0.385	0.410	0.351	0.382	0.621	0.491	0.575	0.548	0.429	0.423	0.422	0.421
	336	**0.362**	**0.379**	0.422	0.432	0.376	0.398	0.521	0.479	0.773	0.631	0.469	0.439	0.456	0.430
	720	**0.416**	**0.412**	0.494	0.472	0.435	0.430	0.571	0.508	0.943	0.716	0.569	0.498	0.554	0.490
	avg	**0.349**	**0.372**	0.414	0.426	0.370	0.393	0.549	0.481	0.686	0.593	0.464	0.441	0.455	0.435
ETTm2	96	**0.159**	**0.247**	0.183	0.279	0.170	0.259	0.212	0.292	0.493	0.476	0.188	0.269	0.192	0.274
	192	**0.217**	**0.287**	0.247	0.320	0.226	0.297	0.297	0.353	0.923	0.658	0.251	0.309	0.246	0.313
	336	**0.269**	**0.322**	0.300	0.353	0.284	0.333	0.328	0.364	1.407	0.822	0.307	0.346	0.301	0.340
	720	**0.357**	**0.377**	0.385	0.408	0.363	0.382	0.456	0.440	1.626	0.905	0.426	0.417	0.400	0.403
	avg	**0.250**	**0.308**	0.279	0.340	0.261	0.318	0.323	0.362	1.112	0.715	0.293	0.335	0.284	0.332
Weather	96	**0.145**	**0.184**	0.189	0.229	0.166	0.217	0.199	0.248	0.230	0.318	0.163	0.215	0.159	0.210
	192	**0.190**	**0.227**	0.239	0.269	0.211	0.257	0.249	0.285	0.357	0.425	0.210	0.254	0.200	0.251
	336	**0.245**	**0.269**	0.294	0.308	0.261	0.296	0.297	0.316	0.464	0.493	0.256	0.292	0.257	0.293
	720	**0.317**	**0.328**	0.366	0.356	0.328	0.342	0.367	0.361	0.515	0.532	0.321	0.339	0.317	0.335
	avg	**0.224**	**0.252**	0.272	0.291	0.242	0.278	0.278	0.303	0.391	0.442	0.238	0.275	0.233	0.272
Electricity	96	**0.135**	**0.226**	0.184	0.276	0.161	0.256	0.279	0.359	0.227	0.334	0.139	0.237	0.143	0.243
	192	**0.150**	**0.240**	0.192	0.284	0.163	0.257	0.282	0.363	0.265	0.366	0.156	0.252	0.159	0.258
	336	**0.166**	**0.258**	0.216	0.308	0.173	0.266	0.289	0.367	0.339	0.417	0.175	0.270	0.170	0.269
	720	**0.205**	**0.290**	0.265	0.347	0.221	0.313	0.333	0.399	0.482	0.478	0.233	0.317	0.230	0.315
	avg	**0.164**	**0.253**	0.214	0.304	0.180	0.273	0.296	0.372	0.328	0.399	0.176	0.269	0.175	0.271
Traffic	96	**0.398**	**0.270**	0.458	0.314	0.421	0.299	0.705	0.386	0.616	0.385	0.414	0.297	0.403	0.293
	192	**0.405**	**0.270**	0.473	0.319	0.439	0.313	0.710	0.393	0.710	0.480	0.426	0.301	0.412	0.295
	336	**0.417**	**0.277**	0.491	0.329	0.448	0.318	0.863	0.456	0.723	0.481	0.434	0.303	0.427	0.316
	720	**0.452**	**0.294**	0.536	0.361	0.478	0.320	0.928	0.485	0.673	0.436	0.487	0.337	0.469	0.325
	avg	**0.418**	**0.278**	0.490	0.331	0.447	0.312	0.801	0.430	0.680	0.446	0.440	0.310	0.427	0.307

A.10.3 ABLATION STUDY RESULTS

Novelty of decomposed frequency learning. Frequency masking is not a new concept, but past approaches randomly mask frequencies of a single time series once (Chen et al., 2023b; Zhang et al., 2023), which show limited forecasting effectiveness due to the lack of common pattern learning from heterogeneous time series that come from various domains. While the multi-frequency masking we proposed randomly mask either high-frequency or low-frequency components of a time series multiple times as the key to enable learning of common time series patterns, such as trends and various long and short term fluctuations. Moreover, different from utilizing frequency masking as a way of data augmentation to enhance the diversity of input data (Chen et al., 2023b; Zhang et al., 2023), we combine multi-frequency masking with reconstruction task as a novel pre-training framework, that learns a universal and unified feature representation by comprehending the data from various frequency perspectives, thereby enabling it to learn generalized representations.

Difference between frequency-domain masking and time-domain noise addition. Multi-frequency masking and reconstruction are not equivalent to the pre-training methods of adding noise and denoising (Noise). Due to the sparsity of time series, the process of adding noise and denoising may potentially disrupt the information of original time series (Dong et al., 2024). In contrast, multi-frequency masking not only preserves the series from such disruption but also helps the model understand temporal patterns from a multi-frequency perspective, thereby helping the model to learn general features better.

Other pre-training tasks. Based on the two points above, we conduct experiments to compare two other pre-training tasks: 1) using frequency-domain augmentation only for data expansion without reconstruction task (*Aug*); 2) replacing multi-frequency masking and reconstruction task with adding time-domian noise and denoise task (*Noise*).As shown in Table 14, we find that ROSE is significantly more effective than *Aug* and *Noise*, which demonstrates the effectiveness of multi-frequency masking and reconstruction task in learning generalized features.

Table 14: Full results of ablation study

Design			Pred_len	ETTm1		ETTm2		ETTh1		ETTh2	
				MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ROSE			96	**0.287**	**0.336**	**0.159**	**0.247**	**0.367**	**0.395**	**0.273**	**0.332**
			192	**0.331**	**0.362**	**0.217**	**0.287**	**0.399**	**0.416**	**0.334**	**0.376**
			336	0.362	**0.379**	**0.269**	**0.322**	**0.405**	**0.423**	**0.358**	**0.397**
			720	**0.416**	**0.412**	**0.357**	**0.377**	**0.416**	**0.443**	**0.376**	**0.417**
			avg	**0.349**	**0.372**	**0.250**	**0.308**	**0.397**	**0.419**	**0.335**	**0.380**
Replace Multi-Frequency Masking		Random Frequency Masking	96	0.330	0.370	0.170	0.262	0.391	0.399	0.303	0.359
			192	0.352	0.392	0.232	0.298	0.406	0.430	0.356	0.395
			336	0.390	0.389	0.276	0.342	0.411	0.432	0.417	0.428
			720	0.452	0.438	0.366	0.392	0.432	0.447	0.420	0.439
			avg	0.381	0.397	0.261	0.324	0.410	0.427	0.374	0.405
		Multi-Patch Masking	96	0.302	0.348	0.168	0.257	0.377	0.408	0.282	0.343
			192	0.336	0.367	0.228	0.297	0.404	0.423	0.343	0.379
			336	0.364	0.385	0.277	0.328	0.405	0.420	0.374	0.403
			720	0.423	0.416	0.364	0.381	0.431	0.455	0.396	0.430
			avg	0.356	0.379	0.259	0.316	0.404	0.426	0.349	0.389
		Patch Masking	96	0.318	0.366	0.168	0.259	0.388	0.412	0.303	0.359
			192	0.355	0.388	0.228	0.298	0.402	0.422	0.370	0.399
			336	0.388	0.406	0.279	0.331	0.411	0.435	0.413	0.428
			720	0.450	0.438	0.370	0.388	0.431	0.459	0.413	0.443
			avg	0.378	0.400	0.261	0.319	0.408	0.432	0.375	0.407
Other pre-training tasks		*Aug*	96	0.304	0.357	0.178	0.266	0.376	0.405	0.281	0.348
			192	0.343	0.379	0.254	0.318	0.409	0.429	0.345	0.389
			336	0.373	0.400	0.299	0.354	0.435	0.453	0.382	0.417
			720	0.444	0.432	0.387	0.408	0.452	0.471	0.434	0.452
			avg	0.366	0.392	0.279	0.336	0.418	0.439	0.360	0.401
		Noise	96	0.303	0.355	0.172	0.261	0.370	0.405	0.280	0.341
			192	0.342	0.376	0.221	0.292	0.403	0.427	0.350	0.384
			336	0.368	0.393	0.272	0.325	0.420	0.439	0.385	0.410
			720	0.423	0.422	0.367	0.386	0.442	0.462	0.403	0.432
			avg	0.359	0.387	0.258	0.316	0.409	0.433	0.355	0.392
w/o		TS-Register	96	0.297	0.345	0.164	0.252	0.379	0.399	0.276	0.336
			192	0.334	0.367	0.221	0.290	0.419	0.420	0.350	0.380
			336	**0.360**	0.384	0.275	0.325	0.438	0.442	0.393	0.411
			720	0.424	0.416	0.364	0.379	0.435	0.448	0.400	0.432
			avg	0.354	0.378	0.256	0.312	0.418	0.427	0.355	0.390
		Prediction Task	96	0.301	0.348	0.166	0.255	0.380	0.407	0.295	0.359
			192	0.343	0.374	0.221	0.291	0.410	0.426	0.372	0.406
			336	0.374	0.393	0.275	0.327	0.440	0.443	0.403	0.429
			720	0.424	0.420	0.366	0.384	0.458	0.476	0.418	0.446
			avg	0.360	0.384	0.257	0.314	0.422	0.438	0.372	0.410
		Reconstruction Task	96	0.329	0.371	0.175	0.265	0.374	0.399	0.296	0.355
			192	0.363	0.391	0.233	0.304	0.407	0.422	0.354	0.389
			336	0.394	0.407	0.287	0.340	0.437	0.440	0.385	0.413
			720	0.461	0.442	0.379	0.396	0.430	0.453	0.408	0.438
			avg	0.387	0.403	0.269	0.327	0.412	0.428	0.361	0.399
From Scratch			96	0.301	0.357	0.171	0.260	0.419	0.439	0.315	0.389
			192	0.358	0.385	0.223	0.294	0.438	0.457	0.366	0.405
			336	0.390	0.396	0.282	0.336	0.484	0.484	0.424	0.435
			720	0.436	0.427	0.366	0.380	0.540	0.538	0.495	0.473
			avg	0.371	0.391	0.261	0.318	0.470	0.480	0.400	0.425

A.10.4 ZERO-SHOT RESULTS

Table 15: Full results in zero-shot setting.

Models		ROSE_512		Timer		MOIRAI		Chronos		TimesFM		Moment	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.382**	0.408	0.414	0.439	0.405	**0.397**	0.494	0.409	0.432	0.405	0.706	0.561
	192	**0.400**	**0.420**	0.440	0.455	0.458	0.428	0.561	0.443	0.492	0.438	0.716	0.579
	336	**0.404**	**0.426**	0.455	0.463	0.509	0.454	0.580	0.460	0.519	0.458	0.705	0.583
	720	**0.420**	**0.447**	0.496	0.496	0.529	0.494	0.605	0.495	0.512	0.477	0.705	0.597
	avg	**0.401**	**0.425**	0.451	0.463	0.475	0.443	0.560	0.452	0.489	0.444	0.708	0.580
ETTh2	96	**0.298**	0.362	0.305	0.355	0.303	**0.338**	0.306	**0.338**	0.311	0.345	0.373	0.416
	192	**0.336**	0.385	0.365	0.406	0.369	**0.384**	0.396	0.394	0.401	0.397	0.384	0.422
	336	**0.353**	**0.399**	0.378	0.413	0.397	0.410	0.423	0.417	0.436	0.430	0.386	0.426
	720	**0.395**	**0.432**	0.414	0.457	0.447	0.450	0.442	0.439	0.437	0.450	0.425	0.454
	avg	**0.346**	**0.394**	0.366	0.408	0.379	0.396	0.392	0.397	0.396	0.405	0.392	0.430
ETTm1	96	0.512	0.460	0.440	0.422	0.660	0.476	0.514	0.443	**0.366**	**0.374**	0.679	0.544
	192	0.512	0.462	0.505	0.458	0.707	0.500	0.608	0.475	**0.413**	**0.401**	0.690	0.550
	336	0.523	0.470	0.570	0.490	0.730	0.515	0.690	0.507	**0.445**	**0.429**	0.701	0.557
	720	0.552	0.490	0.659	0.534	0.758	0.536	0.733	0.555	**0.513**	**0.470**	0.719	0.569
	avg	0.525	0.471	0.544	0.476	0.714	0.507	0.636	0.495	**0.434**	**0.419**	0.697	0.555
ETTm2	96	0.224	0.309	0.203	0.285	0.216	0.282	0.202	0.293	**0.189**	**0.257**	0.230	0.308
	192	0.266	0.333	**0.265**	0.327	0.294	0.330	0.286	0.348	0.277	**0.325**	0.285	0.338
	336	**0.310**	**0.358**	0.319	0.361	0.368	0.373	0.355	0.386	0.350	0.381	0.339	0.369
	720	**0.395**	**0.407**	0.405	0.410	0.494	0.439	0.409	0.425	0.464	0.448	0.423	0.424
	avg	**0.299**	**0.352**	0.360	0.386	0.343	0.356	0.313	0.363	0.320	0.353	0.319	0.360
Weather	96	**0.200**	0.260	0.190	**0.236**	0.188	0.250	0.209	0.244	-	-	0.216	0.271
	192	0.239	0.288	0.261	0.293	**0.237**	**0.284**	0.254	0.288	-	-	0.264	0.306
	336	**0.279**	**0.315**	0.332	0.340	0.282	0.323	0.301	0.332	-	-	0.313	0.336
	720	**0.340**	0.357	0.385	0.381	0.359	**0.345**	0.388	0.374	-	-	0.369	0.380
	avg	**0.265**	0.305	0.292	0.312	0.267	**0.300**	0.288	0.310	-	-	0.291	0.323
Electricity	96	0.209	0.307	0.210	0.312	0.212	0.301	**0.194**	**0.266**	-	-	0.844	0.761
	192	0.219	0.315	0.239	0.337	0.225	0.320	**0.218**	**0.289**	-	-	0.850	0.762
	336	0.236	0.330	0.284	0.372	0.245	0.333	0.244	**0.321**	-	-	0.862	0.766
	720	**0.273**	**0.328**	0.456	0.479	0.282	0.358	0.324	0.371	-	-	0.888	0.774
	avg	**0.234**	0.320	0.297	0.375	0.241	0.328	0.245	**0.312**	-	-	0.861	0.766
Traffic	96	0.572	0.407	**0.526**	**0.368**	-	-	0.562	0.378	-	-	1.390	0.800
	192	0.575	0.406	**0.561**	**0.385**	-	-	0.579	0.412	-	-	1.403	0.802
	336	**0.588**	**0.411**	0.614	0.412	-	-	0.594	0.420	-	-	1.415	0.804
	720	**0.618**	**0.422**	0.749	0.464	-	-	0.723	0.472	-	-	1.437	0.808
	avg	**0.588**	0.412	0.613	**0.407**	-	-	0.615	0.421	-	-	1.411	0.804

A.11 THE DETAILED SETTING OF T-SNE VISUALIZATION

In Figure1(b), we select three datasets (Pems08, PSRA, Electricity) from the transport, nature, and energy domains respectively, and compared the differences in hidden representations between direct transfer and adaptive transfer. Specifically, direct transfer refers to the case where domain specific information is not considered, while adaptive transfer considers domain specific information that is learned by register tokens. We visualize the output of the encoder's hidden representations using t-SNE.

A.12 ADDITIONAL RESULTS

A.12.1 THE RESULTS WITH LOOK-BACK WINDOW $L = 96$

To demonstrate ROSE excels not only with fixed inputs, we evaluate it using input lengths significantly shorter than the 512 employed during pre-training. As shown in Table 16, ROSE still achieves the state-of-the-art performance, indicating the effective transfer of pre-trained knowledge with a shorter look-back window.

Table 16: The results for ROSE and other baselines in full-shot setting with look-back window of 96. The average results of all predicted lengths are listed here.

Models	ROSE		ITransformer		PatchTST		Timesnet		Dlinear		GPT4TS		S^2IP-LLM	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	0.389	**0.389**	0.407	0.410	**0.387**	0.400	0.400	0.406	0.403	0.407	0.389	0.397	0.390	0.399
ETTm2	**0.272**	**0.321**	0.288	0.332	0.281	0.326	0.291	0.333	0.350	0.401	0.285	0.331	0.278	0.327
ETTh1	**0.432**	**0.426**	0.454	0.447	0.469	0.454	0.458	0.450	0.456	0.452	0.447	0.436	0.444	0.431
ETTh2	**0.376**	**0.393**	0.383	0.407	0.387	0.407	0.414	0.427	0.559	0.515	0.381	0.408	0.378	0.402
Weather	**0.257**	**0.276**	0.258	0.278	0.259	0.281	0.259	0.287	0.265	0.317	0.264	0.284	0.266	0.284
Electricity	**0.176**	**0.268**	0.178	0.270	0.205	0.290	0.192	0.296	0.354	0.414	0.205	0.290	0.195	0.285
Traffic	0.440	**0.276**	**0.428**	0.282	0.481	0.304	0.620	0.336	0.625	0.383	0.488	0.317	0.467	0.305

A.12.2 THE SHORT-TERM FORECASTING IN ZERO-SHOT SETTING.

We evaluate ROSE for short-term prediction in zero-shot setting following Moment (Goswami et al., 2024). For extremely short input sequences in the M4 dataset, we adapt them by applying padding. As shown in Table 17, ROSE demonstrates competitive performance compared to other baselines.

Table 17: The results on short-term forecasting in zero-shot setting, measured using SMAPE.

Models	ROSE	Moment	GPT4TS	TimesNet	N-BEATS
M4 Yearly	**14.08**	14.84	14.80	14.40	14.18
M4 Quarterly	11.79	12.02	**11.77**	13.21	12.25
M4 Monthly	15.33	15.80	15.36	15.67	**15.24**

A.12.3 THE COMPARISON WITH THE TIME SERIES REPRESENTATION METHOD

To demonstrate the effectiveness of the pre-trained model in comparison to existing time series representation learning methods, we select recent time series representation learning methods, including SimMTM (Dong et al., 2024), TS2Vec (Yue et al., 2022) and TF-C (Zhang et al., 2022). Table 18 illustrates the performance of ROSE in full-shot and 10% few-shot settings, in comparison with that of time series representation learning methods in full-shot settings. It can be observed that ROSE outperforms the representation learning methods in full-shot setting, and achieves a competitive performance even in 10% few-shot setting, which substantiates the effectiveness of ROSE as a pre-trained model.

Table 18: The results for ROSE in full-shot setting and 10% few-shot setting, compared with other time series representation learning methods in full-shot setting. The average results of all predicted lengths are listed here.

Models	ROSE		ROSE (10%)		SimMTM		Time2Vec		TF-C	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	**0.341**	**0.367**	0.349	0.372	**0.341**	0.377	0.691	0.547	0.732	0.652
ETTm2	**0.246**	**0.305**	0.250	0.308	0.258	0.315	0.316	0.351	1.721	0.922
ETTh1	**0.391**	**0.414**	0.397	0.419	0.401	0.423	0.426	0.436	0.614	0.601
ETTh2	**0.331**	**0.374**	0.335	0.380	0.342	0.384	0.423	0.459	0.387	0.374
Weather	**0.217**	**0.251**	0.224	0.252	0.224	0.262	0.231	0.264	0.286	0.349
Electricity	**0.155**	**0.248**	0.164	0.253	0.161	0.254	0.203	0.283	0.355	0.389
Traffic	**0.390**	**0.264**	0.418	0.278	0.393	0.268	0.450	0.330	0.702	0.443

A.13 Additional Model Analysis

A.13.1 The Analysis of multiple prediction heads

Pre-train: We compared the full-shot performance of using four prediction heads versus a single prediction head in calculating the loss across four different prediction lengths during pre-training. As shown in the Table 19, using four prediction heads consistently yields better performance across all lengths. This phenomenon may be because training four prediction heads in pre-training allows the model to focus on forecasts of different lengths, enhancing accuracy in specific ranges. Conversely, a single prediction head must handle both short-term and long-term forecasts, requiring the model to balance accuracy across various lengths, which could limit its performance on multiple prediction lengths.

Table 19: The results in full-shot setting of pre-training with four prediction heads or a single prediction head in calculating the loss across four different prediction lengths.

Models		ETTh1		ETTh2		ETTm1		ETTm2	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
pre-train_w_four_heads	96	**0.354**	**0.385**	**0.265**	**0.320**	**0.275**	**0.328**	**0.157**	**0.243**
	192	**0.389**	**0.407**	**0.328**	**0.369**	**0.324**	0.358	**0.213**	**0.283**
	336	**0.406**	**0.422**	**0.353**	**0.391**	**0.354**	**0.377**	**0.266**	**0.319**
	720	**0.413**	**0.443**	**0.376**	**0.417**	**0.411**	**0.407**	**0.347**	**0.373**
	avg	**0.391**	**0.414**	**0.331**	**0.374**	**0.341**	**0.367**	**0.246**	**0.305**
pre-train_w_one_head	96	0.357	0.388	0.269	0.325	0.277	0.330	0.158	0.245
	192	0.394	0.411	0.333	0.372	**0.324**	**0.356**	0.215	0.287
	336	0.415	**0.422**	0.359	0.395	0.360	0.380	0.272	0.325
	720	0.420	0.444	0.390	0.425	0.421	0.410	0.355	0.385
	avg	0.397	0.416	0.338	0.379	0.346	0.369	0.250	0.310

Inference: During inference, when the prediction length is covered by multiple heads, we select a prediction head whose output length is the closest to the prediction length. For example, if the prediction length is 48, we select only the prediction head whose output length is 96, even though the other three heads could also perform the prediction length of 48 by cropping.

To demonstrate the effectiveness of this practice, we evaluate the prediction performance for different prediction lengths that are covered by multiple heads, whose results are shown in Table 20 and Table 21. As an example, the first column indicates that using prediction heads of 96, 192, 336, and 720 respectively, to predict with the length of 48. It is evident that the strategy of choosing the closest prediction head allows the model to adapt to various prediction lengths and achieve SOTA performance.

Table 20: The prediction performance of ROSE in ETTh1 for different prediction lengths that are covered by multiple heads.

prediction heads\ prediction lengths	48		96		192		336		720	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
head of 96	**0.325**	**0.364**	**0.354**	**0.385**	-	-	-	-	-	-
head of 192	0.327	0.365	0.358	**0.385**	**0.389**	**0.407**	-	-	-	-
head of 336	0.327	0.365	0.361	0.386	0.399	0.410	**0.406**	**0.422**	-	-
head of 720	0.329	0.365	0.359	0.389	0.401	0.411	0.429	0.425	**0.413**	**0.443**

Table 21: The prediction performance of ROSE in ETTm2 for different prediction lengths that are covered by multiple heads.

prediction heads\ prediction lengths	48		96		192		336		720	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
head of 96	**0.120**	**0.213**	**0.157**	**0.243**	-	-	-	-	-	-
head of 192	**0.120**	**0.213**	0.159	0.245	**0.213**	**0.281**	-	-	-	-
head of 336	0.122	0.214	0.159	0.244	0.216	0.283	**0.266**	**0.319**	-	-
head of 720	0.124	0.216	0.159	0.245	0.216	0.284	0.269	0.321	**0.347**	**0.373**

A.13.2 MULTI-LOSSES BALANCING

To validate ROSE's robustness of the multi-losses, we introduce a hyper-parameter $\lambda \in (0, 1)$, and define $\mathcal{L}_{\text{pretrain}} = \lambda \mathcal{L}_{\text{reconstruction}} + (1 - \lambda)\mathcal{L}_{\text{prediction}} + \mathcal{L}_{\text{register}}$. Since the register loss only constrains the parameter updates of register, its gradient does not influence to the backbone of the model. Therefore, the register loss does not cause an imbalance in the training of the model. We vary λ's value and report the results in full-shot setting. As shown in Table 22, ROSE is not sensitive to changes of λ, thus balancing the loss of the model is not challenging. Therefore, our final loss function does not contain λ.

Table 22: The results in full-shot setting of ROSE pre-trained with different λ. The average results of all predicted lengths are listed here.

λ	0.2		0.4		0.6		0.8		Standard Deviation	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.3973	0.4199	0.3978	0.4193	0.3978	0.4205	0.3996	0.423	0.0008	0.0014
ETTh2	0.3339	0.379	0.3347	0.3802	0.3369	0.3822	0.3351	0.383	0.0011	0.0016
ETTm1	0.3500	0.3733	0.3512	0.3747	0.3492	0.3717	0.3479	0.3719	0.0012	0.0012
ETTm2	0.2538	0.3111	0.2534	0.3095	0.2512	0.3092	0.2505	0.3092	0.0014	0.0007

A.13.3 THE ANALYSIS OF DECOMPOSED FREQUENCY LEARNING

To demonstrate the effectiveness of decomposed frequency learning in capturing unified representations, we pre-trained the model using multi-frequency masking, patch masking and multi-patch masking. We visualize the reconstruction performance of the three methods in out-of-distribution (OOD) scenarios. As shown in Table 16, the model pre-trained with multiple frequency masking exhibits greater robustness to complex temporal patterns, confirming that decomposed frequency learning can assist in learning unified representations.

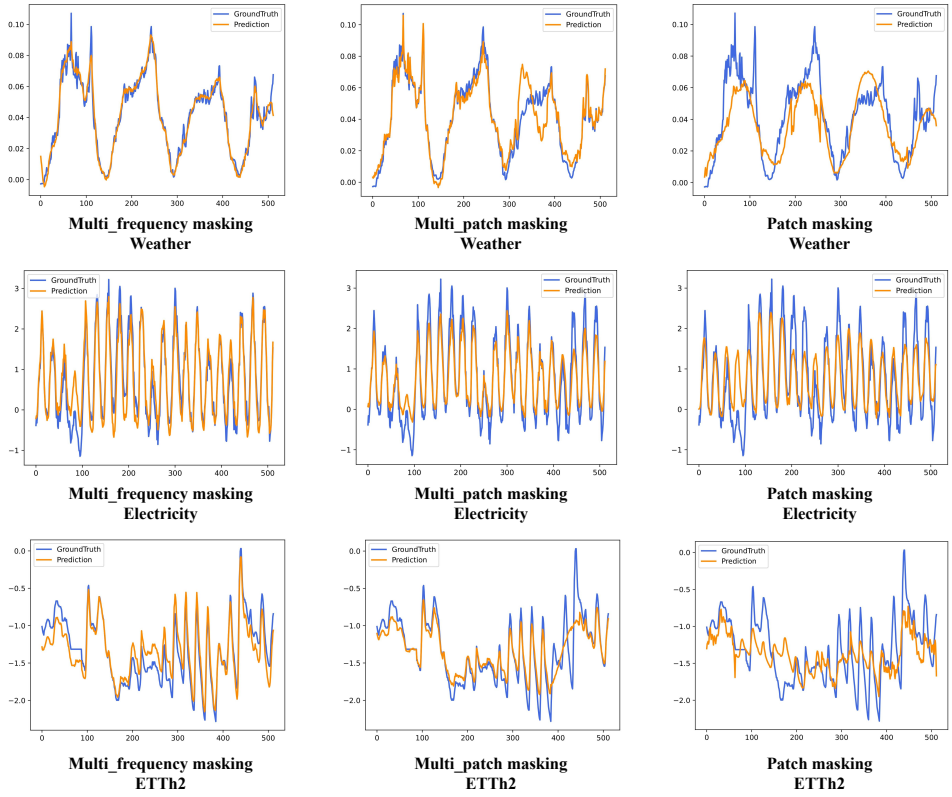


Figure 16: The visualization of the reconstruction performance of models pre-trained with three methods (multi-frequency masking, patch masking and multi-patch masking) respectively in out-of-distribution (OOD) scenarios.