# SEQFUSION: Sequential Fusion of Pre-Trained Models for Zero-Shot Time-Series Forecasting

Ting-Ji Huang, Xu-Yang Chen, Han-Jia Ye

**Abstract**—Unlike traditional time-series forecasting methods that require extensive in-task data for training, *zero-shot* forecasting can directly predict future values given a target time series without additional training data. Current zero-shot approaches primarily rely on pre-trained generalized models, with their performance often depending on the variety and relevance of the pre-training data, which can raise privacy concerns. Instead of collecting diverse pre-training data, we introduce SEQFUSION in this work, a novel framework that *collects and fuses diverse pre-trained models (PTMs) sequentially* for zero-shot forecasting. Based on the specific temporal characteristics of the target time series, SEQFUSION selects the most suitable PTMs from a batch of pre-collected PTMs, performs sequential predictions, and fuses all the predictions while using minimal data to protect privacy. Each of these PTMs specializes in different temporal patterns and forecasting tasks, allowing SEQFUSION to select by measuring distances in a shared representation space of the target time series with each PTM. Experiments demonstrate that SEQFUSION achieves competitive accuracy in zero-shot forecasting compared to state-of-the-art methods. Code is available at https://github.com/Tingji2419/SeqFusion .

**Index Terms**—time series forecasting, zero-shot, pre-trained model.

## 1 INTRODUCTION

Time-series forecasting is essential in various fields such as healthcare [1], finance [2], and environmental science [3], where it supports decision-making by predicting future trends based on historical data. Current predominant approaches, including traditional statistical methods [4], [5] and modern deep learning techniques [6]–[13], often require extensive in-task training data to capture complex temporal patterns, which are tailored to a particular field or environment similar to the target time series [14]. However, in many real-world scenarios, there is often little or even no available training data [15]–[17], which limits the effectiveness of these traditional and deep learning methods, reducing their practicality in real-world applications.

To address these challenges, *zero-shot* forecasting has emerged as a promising solution [18]. This approach enables reliable predictions of target time series without in-task training data. Current studies mainly employ generalized models pre-trained on diverse datasets [10], [19], [20], which require the centralized collection of large and diverse data. However, these generalized models typically demand substantial memory storage and raise significant data privacy concerns [21], [22]. Given that sharing pre-trained models (PTMs) or small demo datasets instead of entire source data is a common strategy for protecting data privacy [23], we pioneer an approach that utilizes a set of lightweight PTMs to help for zero-shot forecasting.

*Ting-Ji Huang, Xu-Yang Chen, and Han-Jia Ye are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210023, China. E-mail: huangtj, chenxy, yehj@lamda.nju.edu.cn.*

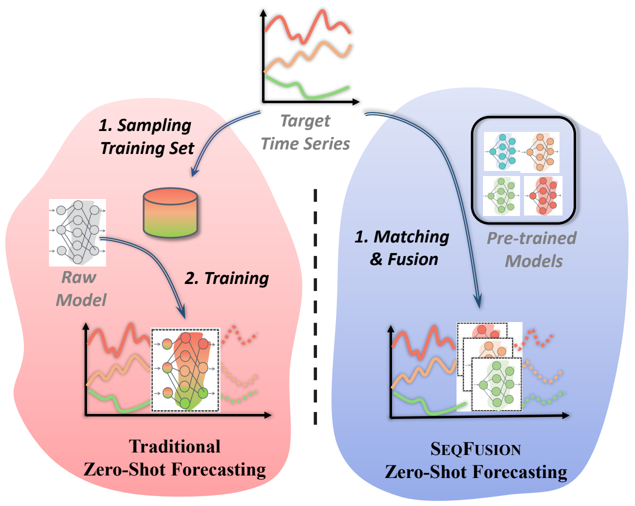


Fig. 1. Comparison of traditional zero-shot forecasting and SEQFUSION. Traditional methods rely on sampling data and training, while SEQFUSION leverages pre-trained models (PTMs) through matching the target time series to suitable PTMs and fusing their predictions.

In this work, we propose SEQFUSION, a novel method that leverages pre-trained models (PTMs) for zero-shot time-series forecasting. Instead of collecting diverse data, we create a *model zoo* containing a wide range of PTMs, mimicking the diverse set of PTMs that would naturally emerge from various time-series forecasting tasks in real-world applications [24]. Each PTM in our model zoo is pre-trained on distinct datasets from various tasks, enabling them to specialize in different temporal patterns. Model zoo is common and acceptable in real-world applications [25], [26], such as models for stocks prediction [27], where a company may have trained a large number of deep models for different time periods, different regions, or different domains. When faced with a new forecasting task, we would like to utilize these deep-learning PTMs if there is a lack of datasets.

Collecting PTMs is beneficial for data privacy protection, as it reduces the risk of exposing sensitive information in the time series. Since PTMs are often black-box models, they do not reveal the underlying data used for their training.

Thus, our approach provides the necessity and feasibility of sharing time-series PTMs in the future, which has already been a great success in the field of vision and natural language process [21], [28]. Furthermore, storing PTMs reduces storage costs compared to storing raw time series data. PTMs require less storage space, and they can be easily shared and reused across different tasks and applications.

SEQFUSION employs a diverse collection of PTMs and a dynamic selection mechanism to identify the most suitable PTMs based on the specific characteristics of each variate in the target time series. With the selected PTMs, SEQFUSION forecasts sequentially and fuses the outputs of each PTM for every variate to obtain the final predictions. Each of these PTMs is pre-trained on different and diverse data, thus specializing in different temporal patterns and forecasting tasks. For instance, a model pre-trained on the Hospital [29] dataset might effectively forecast certain variates in the Illness [1] dataset due to their shared medical data characteristics and similar recording frequencies (See Table 7 for more details). This flexibility is particularly useful for single-variate scenarios, where PTMs can be used interchangeably if the history length of two forecasting tasks is consistent, enhancing the practicality of the forecasting process.

The success of SEQFUSION hinges on two key factors: obtaining accurate characteristics of the target time series and the PTMs and aligning these characteristics to match suitable models with the target series. This is achieved through a time-series feature extractor, trained via an unsupervised learning process, ensuring that time series from the same dataset have similar representations, while also aligning PTMs' representations to reflect their transferability across related datasets. This dual alignment process is vital for ensuring that the selected PTMs are highly relevant and capable of producing accurate predictions. Using only a few data that do not harm privacy, we extract these representations and match appropriate PTMs with the target time series based on representation distance.

Once suitable models are selected, SEQFUSION employs a sequential forecasting mechanism that predicts values segment by segment for each variate, and allows for aggregated predictions of the most relevant few PTMs. This direct prediction method, without additional training data, ensures flexibility and adaptability to various time series. Figure 1 illustrates the contrast between traditional forecasting, which relies on sampling from an in-task training set, and our method, which efficiently utilizes a set of PTMs without such sampling. Our contributions are:

- We introduce SEQFUSION, a novel framework that leverages pre-trained models (PTMs) for effective zero-shot time-series forecasting, eliminating the need for large-scale pre-training data.
- SEQFUSION selects suitable PTMs based on the target time series characteristics, performs sequential predictions, and fuses the predictions of all the selected PTMs to produce a robust final forecast.
- Comprehensive experiments across diverse datasets demonstrate that SEQFUSION achieves competitive accuracy to state-of-the-art methods, particularly in environments with limited data and storage space.

1. https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

## 2 RELATED WORK

**Time-Series Forcasting**. Time-series forecasting plays a vital role in various domains such as finance [2], weather prediction [3], and industrial manufacturing [14], [30], [31]. Classic methods like ARIMA [4] have long been the cornerstone of time-series forecasting. In recent years, the advent of deep learning methods has become popular in this area. Many methods utilize deep sequence models in an attempt to capture temporal dependencies and long-range dependencies in the series [7], [32], [33]. Some recent studies leverage the large language models for time-series analysis that are based on billions of parameters and tremendous training data from various domains [10], [19]. However, these deep learning methods often require large amounts of relevant training data to achieve great performance and are difficult to perform in some data-lacking applications [34], [35].

**Unsupervised Representation of Time Series**. Unsupervised representation learning of time series aims to extract generalized representations from complex raw time series data without human supervision. Previous works apply joint clustering and feature learning approaches and show remarkable performance [36], while others adopt deep models such as auto-encoders or sequence-to-sequence models to generate time-series representations [37], [38]. Recently, self-supervised learning has emerged as a new paradigm, which leverages pretext tasks to autonomously generate labels by utilizing intrinsic information derived from the unlabeled data [39]–[42]. These inspire us to extract features of target time series for similar sequence matching to help forecasting in data-limited prediction tasks.

**Zero-Shot Forecasting**. Zero-shot forecasting addresses the challenge of predicting unseen or novel time-series sequences without historical data. This paradigm shift is crucial for scenarios where traditional forecasting methods are impractical due to the absence of relevant training data [35]. Early works like Meta-N-BEATS [18] employ a model based on N-BEATS [43] with flexible basis functions to generalize across different time-series domains. ForecastPFN [34] predicts future series by constructing synthetic data and training a model with this, bypassing the need for real-world training data. Meanwhile, some previous work utilized pre-trained language models to generate predictions based on contextual understanding of time-series data [10], [19], [35], [44], [45]. These methods leverage the inherent ability of language models to understand and generate sequences, thereby enabling them to make informed predictions even in the absence of direct historical data [20], [46]. However, the good performance of these zero-shot forecasting methods typically requires large and diverse pre-training data. In this work, we explore collecting diverse PTMs instead of pre-training data for zero-shot forecasting.

**Pre-Trained Models for Forecasting**. Leveraging pre-trained models (PTMs) for forecasting tasks has gained momentum, especially as it allows models to generalize across diverse tasks with minimal task-specific training. In time-series contexts, PTMs can transfer learned temporal patterns from one domain to another, improving forecasting accuracy when data is sparse or task-specific patterns are hard to capture. Several recent studies demonstrate the efficacy of PTMs in forecasting [10], [20], [34]. TimeGPT [19], a large-scale, closed-source model which is pre-trained on diverse time-series datasets. It captures complex temporal dependencies, making it highly effective in zero-shot forecasting scenarios. Chronos [20] tokenizes time-series data through scaling and quantization and uses transformer-based architectures to process these sequences. Moirai [46], a masked encoder-based transformer pre-trained on a large-scale dataset, which includes over 27 billion observations across nine domains. Its extensive pre-training enables Moirai to handle with an arbitrary number of variates for multivariate time series, and perform competitively in zero-shot forecasting compared with fully supervised models. TimesFM [47] employs a patched-decoder attention mechanism pre-trained on a large time-series corpus to capture long-range dependencies. This structure allows TimesFM to recognize nuanced temporal patterns, making it suitable for tasks requiring detailed temporal analysis. These works underline the potential of PTMs to serve as a shared repository or model zoo, where models trained on various domains and tasks can be selectively utilized for new forecasting applications, reducing the reliance on large datasets and extensive retraining. In this work, we build on the model zoo concept to facilitate forecasting in data-limited settings, aiming to make time-series PTMs more accessible and effective across various forecasting domains.

## 3 Preliminary for SeqFusion

We first describe the multivariate time-series forecasting problem under the zero-shot situation. Then we discuss a solution with PTMs and the sequential forecasting method used in SeqFusion.

### 3.1 Multivariate Zero-Shot Forecasting

In multivariate time-series forecasting under the zero-shot setting, we are given a historical time series $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_C\} \in \mathbb{R}^{T\times C}$ with $C$ variates of a limited length $T$. The objective is to predict the future values of each variate $\mathbf{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_C\} \in \mathbb{R}^{H\times C}$ over a horizon $H$ through a model $\boldsymbol{\phi} : \mathbb{R}^{T\times C} \rightarrow \mathbb{R}^{H\times C}$, where the length $T$ is insufficient to train a deep model even for one step [18].

**Algorithm 1** Sequential Forecasting Process

```
1:  Input: Time series data $\mathbf{x}$, model $\phi$, input length $T$, forecast horizon $H$, iteration steps $n$
2:  Output: Forecast $\hat{\mathbf{y}}$ over horizon $H$
3:  Initialize $\mathbf{x}_0 = \mathbf{x}$
4:  Compute initial output $\mathbf{y}^0 = \phi(\mathbf{x}^0)$
5:  for $t = 1$ to $n$ do
6:      Concatenate inputs and previous outputs: $[\mathbf{x}^0, \mathbf{y}^0, \ldots, \mathbf{y}^{t-1}]$
7:      Trim concatenated data to maintain length $T$: $\mathbf{x}^t = \text{Trim}([\mathbf{x}^0, \mathbf{y}^0, \ldots, \mathbf{y}^{t-1}])$
8:      Apply model $\phi$ to get new output: $\mathbf{y}^t = \phi(\mathbf{x}^t)$
9:  end for
10: Concatenate all outputs: $[\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^n]$
11: Trim concatenated outputs to match forecast horizon $H$: $\hat{\mathbf{y}} = \text{Trim}([\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^n])$
```

### 3.2 Sequential Forecasting with PTMs

When dealing with a target time-series forecasting task with limited input length, direct training a deep model may not be feasible due to data limitations. Instead of collecting diverse data to pre-train a generalized model, a feasible approach is directly utilizing a PTM to make zero-shot forecasting.

Given a target time-series $\mathbf{X}$ and a PTM $\boldsymbol{\phi} : \mathbb{R}^{T\times C} \rightarrow \mathbb{R}^{h\times C}$ pre-trained on another task, if the length of $\mathbf{X}$ is the same as the input history length $T$ of $\boldsymbol{\phi}$ and the number of variates is also the same, we can use $\boldsymbol{\phi}$ to directly predict the future value $\hat{\mathbf{Y}} = \boldsymbol{\phi}(\mathbf{X})$. However, considering that the prediction length of the PTM $\boldsymbol{\phi}$ is often fixed to $h$, which is common in recent deep learning methods [6], [32], [48], it may not be directly applicable in a zero-shot setting, especially when the required horizon $H$ is longer than the prefix prediction length $h$.

To address this, we introduce a sequential forecasting approach for $n$-step ahead recursive forecasting [49], [50]. Specifically, for each variate, the task of forecasting a horizon $H$ is decomposed into $n = \lceil H/h \rceil$ forecasting blocks. For each block, we make multivariate predictions using $\boldsymbol{\phi}$, where the output of the previous module is truncated to a suitable input length for the model and used as input for the next module. Ultimately, we synthesize the output of each module and truncate it to $H$. Each yields a predicted value of $\hat{\mathbf{y}}_c$ for each variate. The steps of this sequential prediction are detailed in Algorithm 1.

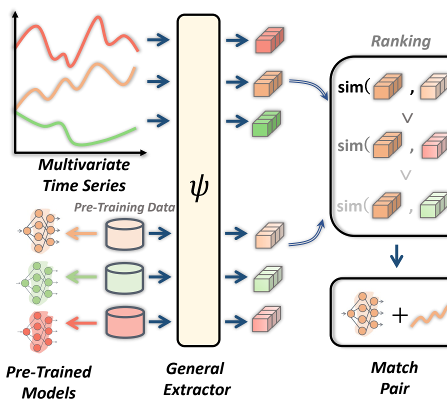
## 4 SeqFusion

In this section, we first introduce the model zoo with diverse PTMs and the overall architecture of SeqFusion for zero-shot time-series forecasting. We then describe the process of obtaining time-series representations, as well as incorporating time-series transferability to improve the semantics of the representations. Following this is the matching mechanism that allows for PTM selection. Finally, we introduce how SeqFusion can benefit from the fusion of multiple most suitable PTMs for forecasting. The overall architecture of SeqFusion is shown in Figure 2.

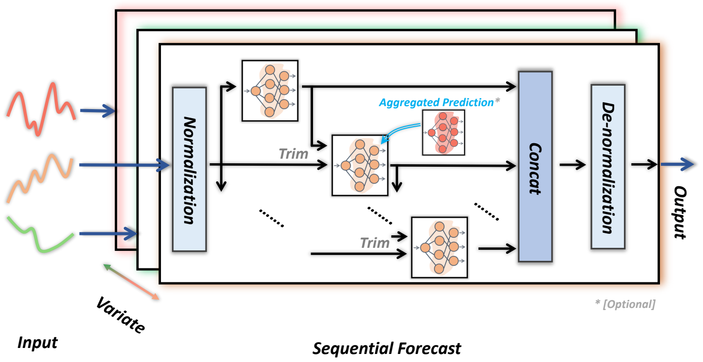
### 4.1 Leveraging PTMs for Time-Series Forecasting

In SeqFusion, we propose to collect diverse PTMs and fuse their sequential predictions for multivariate zero-shot forecasting. Instead of collecting a large amount of diverse data, we create a model zoo containing PTMs pre-trained on various tasks. Specifically, we collect a model zoo $\mathcal{M} = \{\boldsymbol{\phi}_m\}_{m=1}^{M}$ containing diverse PTMs, where each PTM $\boldsymbol{\phi}_m : \mathbb{R}^{T\times 1} \rightarrow \mathbb{R}^{H\times 1}$ is an one-variate model that pre-trained on a different dataset $\mathbf{X}_{\boldsymbol{\phi}_m}$. Given a PTM $\boldsymbol{\phi}_m$, we can directly predict the future value $\hat{\mathbf{y}}_c = \boldsymbol{\phi}_m(\mathbf{x}_c)$ based on the corresponding variate of its history $\mathbf{x}_c \in \mathbf{X}$ once the input length $T$ is consistent.

Once the model zoo $\mathcal{M}$ is created, SeqFusion selects the most suitable PTMs for each variate of the target time series $\mathbf{X}$. We will discuss this selection process later. With the selected PTMs $\{\hat{\boldsymbol{\phi}}_c\}_{c=1}^{C}$ for each variate of the target time series $\mathbf{X}$, the forecasting process is executed through the following stages as shown in Figure 2b:



(a) Matching PTMs for Time Series.



(b) Zero-Shot Forecasting with SEQFUSION.

Fig. 2. Overview of SEQFUSION. (a) SEQFUSION collects diverse PTMs and selects the most suitable PTMs based on the characteristics of the target time series. This selection is based on the representations obtained from a general extractor, to provide a vector for measuring the similarity between the PTMs and each variate in the target time series. (b) SEQFUSION employs sequential forecasting with the selected PTMs for each variate, and fuses all predictions over all variates to generate final forecasts. The sequential forecasting process includes normalization, trimming, and the optional aggregated prediction of the most suitable PTMs, followed by concatenation and de-normalization to produce the final forecasts.

**Normalization**. Each variate $\mathbf{x}_c$ of the input time series $\mathbf{X}$ is first normalized to ensure uniformity in data scale and distribution: $\mathbf{x}_c^{\text{norm}} = (\mathbf{x}_c - \mu_{\mathbf{x}_c})/\sigma_{\mathbf{x}_c}$, where $\mu_{\mathbf{x}_c}$ and $\sigma_{\mathbf{x}_c}$ are the mean and standard deviation of variate $\mathbf{x}_c$. Note that the normalization module decreases the distributional discrepancy among each input variate, making the distribution of the model input more stable.

**Sequential Fusion**. Following normalization, the process employs a sequential forecasting loop, where the selected pre-trained model $\hat{\phi}_c$ is applied repeatedly over $n = H/h$ times for each variate with the initial input $\mathbf{x}_c^{\text{norm}}$, and each block involves a trim operation that refines the output time series for the next iteration in Alg. 1. From here, we get the predictions $\hat{\mathbf{y}}_c^{\text{norm}}$ of every variate.

**Concatenation and De-normalization**. The final outputs from the sequential forecasting steps are concatenated and then de-normalized to convert them back to the original scale of the data:

$$\hat{\mathbf{Y}} = \{\hat{\mathbf{y}}_1, ..., \hat{\mathbf{y}}_C\}, \;\; \hat{\mathbf{y}}_c = \sigma_{\mathbf{x}_c} \cdot \hat{\mathbf{y}}_c^{\text{norm}} + \mu_{\mathbf{x}_c}. \tag{1}$$

The final output of SEQFUSION, $\hat{\mathbf{Y}}$, is a collection of multivariate sequences, each of which represents the predictions for a variate within the specified prediction horizon $H$. This approach not only takes advantage of the PTMs but also implements variate-length prediction through a recursive strategy that can be used to predict any number of independent variates.

### 4.2 Universal Representations for Selection

In SEQFUSION, We select suitable PTMs for the target time series based on the representations of target time series and PTMs. We encode the target time series and PTMs using a general extractor model $\psi$. The model $\psi$ is well-trained on a general dataset $\mathcal{D}$ that does not contain overlapping data with the target time series $\mathbf{X}$. We constructed this generalized dataset $\mathcal{D}$ by randomly sampling sample sub dataset $\mathcal{D}_i$ from the pre-training datasets $\{\mathbf{X}_{\phi_m}\}_{m=1}^{M}$ of the model zoo. And we extract the representation of $\mathbf{X}_{\phi_m}$ to characterize each PTM $\phi_m$ in the model zoo.

**General Extractor.** The general extractor model $\psi$ is trained using an unsupervised learning process inspired by [42], employing an encoder-decoder architecture. The encoder processes input time series to a latent representation $\mathbf{E}(\cdot)$, while the decoder reconstructs the original time series from this latent representation. The output of the encoder, $\mathbf{E}(\cdot)$, is utilized as the representation of the input sequence.

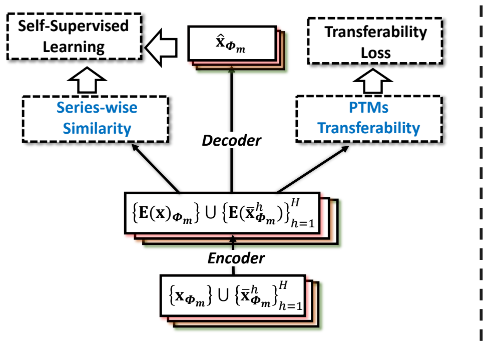
To further enhance the quality of the representations, as shown in Figure 3, the training process includes two key components. Firstly, a series-wise similarity objective aligns representations of time series sampled from the same dataset, ensuring consistency within related data. Given a set of input series $\mathbf{x}_{\phi_m}$ (here we use PTMs' pre-training datasets), we generate multiple masked time series set $\overline{\mathbf{x}}_{\phi_m}$ by randomly masking a portion of time points along the temporal dimension. The original time series and its masked series will present close representations and be far away from the representations from other series, where we use $\text{sim}(\cdot,\cdot)$ to denote the cosine similarity:

$$\mathcal{L}_{\text{constraint}} = -\sum_{s\in\mathbf{x}_{\phi_m}} \left( \sum_{s'\in\overline{\mathbf{x}}_{\phi_m}} \log \frac{\exp(\text{sim}(s,s'))}{\sum_{s''\in\mathbf{x}_{\phi_m}} \exp(\text{sim}(s,s''))} \right). \tag{2}$$

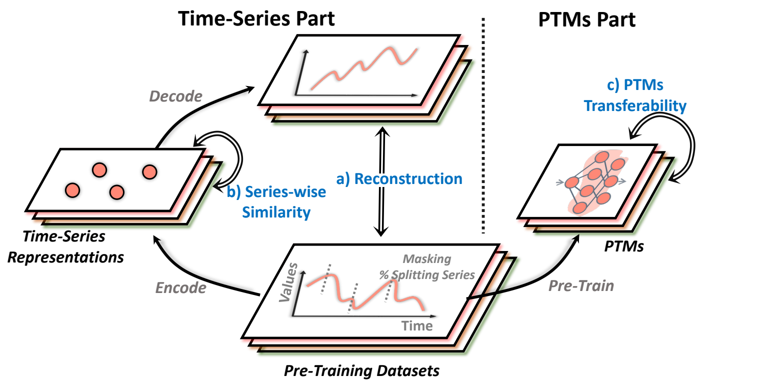
Secondly, the transferability loss aligns the learned representations with the transferability potential of PTMs. Representations learned through reconstruction are inherently time-series-specific and is model-independent, whereas the ability of different PTMs to work on the same training set is likely to be different. in order to better fit the PTM selection mechanism, we introduce the *transferability loss* function. Specifically, given two series $\mathbf{x}_i \in \mathcal{D}_i$, $\mathbf{x}_j \in \mathcal{D}_j$, the model $\psi$ is trained with an additional supervised loss:

$$\mathcal{L}_{\text{trans}} = \sum_{\mathbf{x}_i,\mathbf{x}_j\in\mathcal{D}} \left\| g_{i,j} - \text{sim}\big(\mathbf{E}(\mathbf{x}_i), \mathbf{E}(\mathbf{x}_j)\big) \right\|_2^2, \tag{3}$$

where $g_{i,j}$ is the transferability from dataset $\mathcal{D}_i$ to dataset $\mathcal{D}_j$. In our experiments, the $g_{i,j}$ is computed using the 1-MSE metric, which measures the MSE metric performance


Self-Supervised Learning
$\hat{\mathbf{x}}_{\phi_m}$
Transferability Loss
Series-wise Similarity
Decoder
PTMs Transferability
$\{\mathbf{E}(\mathbf{x})_{\phi_m}\} \cup \{\mathbf{E}(\bar{\mathbf{x}}^h_{\phi_m})\}_{h=1}^H$
Encoder
$\{\mathbf{x}_{\phi_m}\} \cup \{\bar{\mathbf{x}}^h_{\phi_m}\}_{h=1}^H$

(a) Architecture of General Extractor.



(b) Training process of General Extractor.

Fig. 3. (a) Architecture of the General Extractor. The encoder-decoder model extracts time-series representations optimized through self-supervised learning based on series-wise similarity and transferability loss to align time-series representations with PTMs' transferability. (b) Training Process of the General Extractor. In the Time-Series part, raw time-series data from pre-training datasets are masked and split to create input pairs. The encoder generates time-series representations while the decoder reconstructs the input. A series-wise similarity loss aligns representations of series from the same dataset and allows representations from different datasets to be far apart. In the PTMs Part, PTMs are evaluated for transferability using metrics like 1-MSE, and this information is integrated to refine representations, ensuring they reflect both the intrinsic characteristics of the time series and the ability of PTMs to generalize to related tasks.

of PTMs trained on $\mathcal{D}_i$ and tested on $\mathcal{D}_i$. By aligning the similarity of representations with transferability scores, the extractor $\psi$ learns representations that not only capture time-series features but also reflect the potential utility of PTMs across datasets. Finally, the overall optimization process of the general extractor can be represented as follows:

$$\min_{\Theta} \left\|\mathbf{x}_{\phi_m} - \hat{\mathbf{x}}_{\phi_m}\right\|_2^2 + \mathcal{L}_{\text{trans}} + \lambda \mathcal{L}_{\text{constraint}}, \tag{4}$$

where the first part is the reconstruction loss, $\hat{\mathbf{x}}_{\phi_m}$ denotes the reconstructed output of the decoder, and $\Theta$ denotes the set of all parameters of the general extractor. This design ensures that the representations of PTMs pre-trained on related datasets are closely aligned in the embedding space, while representations of dissimilar datasets remain distinct. Consequently, models trained on datasets that are similar to the target time series are assigned higher similarity scores (see Figure 4a for more details), facilitating accurate and efficient PTM selection.

**Representation Extraction.** We extract the variate-wise representations the target time-series $\mathbf{X}$ and the representations of PTMs $\{\phi_m\}_{m=1}^M$ via the general extractor $\psi$. Specifically, for each variates $\mathbf{x}_c$ of the target time series $\mathbf{X}$, we encode them separately, denoted by $\boldsymbol{\mu}_c \in \mathbb{R}^d$; for each PTM $\phi_m$, we sampled a batch of data $\phi_m$ from its pre-training dataset $\mathbf{X}_{\phi_m}$, and use the average encoding $\boldsymbol{\theta}_m \in \mathbb{R}^d$ of the sampled data as the representation:

$$\boldsymbol{\mu}_c = \psi(\mathbf{x}_c)\big|_{c=1}^C, \quad \boldsymbol{\theta}_m = \psi(\mathbf{x}_{\phi_m})\big|_{m=1}^M. \tag{5}$$

With this form of representation, we project the representation of the model and the representation of the target time series into the same space. Models trained from related datasets are similar in this representation, and models with pre-trained datasets that are similar to the target time series data are similar in this representation. This crucial step makes our subsequent model-matching process feasible.

## 4.3 Matching PTMs for Target Time Series

We extract the variate-wise representations $\{\boldsymbol{\mu}_c\}_{c=1}^C$ of the target time series $\mathbf{X}$ and the representations $\{\boldsymbol{\theta}_m\}_{m=1}^M$ of PTMs $\{\phi_m\}_{m=1}^M$ via the general extractor $\psi$. We then select PTMs for each variate of $\mathbf{X}$ based on the similarity between the target time-series representation $\{\boldsymbol{\mu}_c\}_{c=1}^C$ and those $M$ PTMs' representations $\{\boldsymbol{\theta}_m\}_{m=1}^M$. We expect that the higher the similarity, the more helpful a PTM is for the target time-series. We use the cosine similarity $\mathrm{sim}(\cdot,\cdot)$ for measuring the similarity, and $\{\hat{\phi}_c\}_{c=1}^C$ to represent the selected PTMs for the target time-series with $C$ variates:

$$\hat{\phi}_c = \operatorname*{argmax}_{\phi_i \in \mathcal{M}} \mathrm{sim}\left(\boldsymbol{\theta}_i, \boldsymbol{\mu}_c\right), \qquad c = 1, ..., C. \tag{6}$$

Given the target time-series $\mathbf{X}$, we now select a PTM set $\{\hat{\phi}_c\}_{c=1}^C$ for each variate using Equation 6. This selection is based on the historical data and specific characteristics of each variate, ensuring that each model is optimized for the type of data it will process.

Since each PTM in the model zoo specializes in a different temporal pattern and prediction task, so given any new time series task, SEQFUSION has the opportunity to select it to help with zero-shot prediction, as long as there is a PTM that has been trained on a similar task.

## 4.4 Optional Aggregated Predictions

To further enhance the accuracy and robustness of the forecasts, SEQFUSION includes an optional aggregated prediction phase. This phase integrates the predictions of $k$ PTMs, selected based on their similarity to the input variate $\mathbf{x}_c$ from Equation 6. Specifically, in each block of the sequential forecasting, we give an average prediction based on the top-$k$ suitable PTMs: $\hat{\mathbf{y}}_c^{\text{aggr}} = \sum_{i=1}^k \hat{\mathbf{y}}_{c,i}^{\text{norm}}/k$, where we use $\hat{\mathbf{y}}_{c,i}^{\text{norm}}$ to represent the prediction from one PTM, and the final predictions are:

$$\hat{\mathbf{Y}}_{\text{aggr}} = \{\hat{\mathbf{y}}_1^{\text{aggr}}, ..., \hat{\mathbf{y}}_C^{\text{aggr}}\},\ \hat{\mathbf{y}}_c^{\text{aggr}} = \sigma_{\mathbf{x}_c} \cdot \hat{\mathbf{y}}_c^{\text{aggr}} + \mu_{\mathbf{x}_c}. \tag{7}$$

Incorporating aggregated predictions into the SeqFusion framework enhances the reliability and accuracy of the final forecast (see Figure 5). This phase ensures that the model can leverage multiple sources of knowledge, making it more robust in various forecasting scenarios.

## 5 Experiments

We evaluate SeqFusion on three zero-shot benchmarks: multivariate datasets with a) lightweight PTMs or b) large language models, and c) univariate datasets with lightweight PTMs. We then analyze the influence of key components in SeqFusion.

### 5.1 Evaluation on Multivariate Time Series

**Setups**. We follow the setting in previous works [32], [34] and evaluate models on seven popular, real-world datasets across various domains, which are summarized in Table 1. The datasets include:

- **ETT (Electricity Transformer Temperature)** [6][2]: This dataset comprises two hourly-level datasets (ETTh) and two 15-minute-level datasets (ETTm). Each dataset contains seven oil and load features of electricity transformers from July 2016 to July 2018.
- **Traffic**[3]: This dataset describes road occupancy rates and contains hourly data recorded by sensors on San Francisco freeways from 2015 to 2016.
- **Electricity**[4]: This dataset collects the hourly electricity consumption of 321 clients from 2012 to 2014.
- **Exchange-Rate** [51][5]: This dataset collects the daily exchange rates of 8 countries from 1990 to 2016.
- **Weather**[6]: This dataset includes 21 indicators of weather, such as air temperature, and humidity, recorded every 10 minutes for year 2020 in Germany.
- **ILI**[7]: This dataset describes the ratio of patients seen with influenza-like illness to the total number of patients. It includes weekly data from the Centers for Disease Control and Prevention of the United States from 2002 to 2021.

Following previous work [34], the look-back window length is set to 36 and the prediction horizon is set to $\{6, 8, 14, 18, 24, 36, 48\}$ for all datasets. We measure the performance using Mean Squared Error (MSE).

$$\text{MSE} = \frac{1}{H}\sum_{i=1}^{H}(\mathbf{Y}_i - \hat{\mathbf{Y}}_i)^2 \tag{8}$$

where we assume that $\mathbf{Y}, \hat{\mathbf{Y}} \in \mathbb{R}^{H\times C}$ are the ground truth and prediction results of the future with $H$ time points and $C$ variates. $\mathbf{Y}_i$ denotes the $i$-th future time point.

**Baselines**. To provide a comprehensive evaluation, we include a variety of forecasting models:

- Naive models: **Last**, which uses the last observed value for future predictions; **Mean**, which predicts future values as the mean of past observations; and **SeasonalNaive** [4] (with a period of 7), which repeats the values from the same period in the previous cycle.
- Statistical methods: we include **ARIMA** [4], known for its autoregressive integrated moving average properties, and **Prophet** [5], a model developed by Facebook designed for handling seasonality and trends in time series data.
- Deep learning models: **Transformer** [52], renowned for its attention mechanism; **Autoformer** [32], which introduces decomposition blocks to enhance the representation of seasonal patterns; **Informer** [6], optimized for efficiency and scalability in handling large datasets; **DLinear** [48], focusing on decomposed linear forecasting; **PatchTST** [8], which leverages patch-based temporal patterns; and **iTransformer** [7], an inverted Transformer designed for multivariate time series.
- Zero-shot forecasting models: **Meta-N-BEATS** [18], which adapts the N-BEATS architecture for meta-learning; **ForecastPFN** [34], a synthetically-trained zero-shot forecastor; and **GPT4TS** [10], a variant of GPT-2 fine-tuned for time series forecasting. We use Meta-N-BEATS and GPT4TS pre-trained on M4 for multivariate forecasting bennchmark.

| Dataset | Channels | Timesteps | Granularity | Domain |
|---|---|---|---|---|
| ETTh1, ETTh2 | 7 | 17,420 | Hourly | Electricity |
| Exchange-Rate | 8 | 7,588 | Daily | Economy |
| Weather | 21 | 52,696 | 10min | Weather |
| ECL | 321 | 26,304 | Hourly | Electricity |
| Traffic | 862 | 17,544 | Hourly | Transport |
| ILI | 7 | 966 | Weekly | Health |

TABLE 1
Statistics of multivariate time series datasets.

The deep learning methods are allowed to train on 50 in-task time stamps as training data, even though this is not a fair comparison. Others see only the input of length 36 and then predict the future. We reproduce all the deep learning methods and zero-shot ones using the officially provided codes.

**Implementation Details**. Our focus is on how PTMs can be utilized to help with downstream time series forecasting tasks. To demonstrate the superiority of our approach, we first collect several super-lightweight PTMs (only around 0.05 MB to 2.31 MB per PTMs), to ensure the memory cost of these models (DLinear and PatchTST) are much smaller than storing the pre-training datasets. Specifically, for multivariate forecasting tasks, we collect 10 PTMs pre-trained on the random sampling set from M3 (3 PTMs), M4 (5 PTMs) and Tourism (2 PTMs) based on the PatchTST [8] architecture. These datasets are sampled at frequencies ranging from minutes to months, with a variety of domains and covering frequencies common to time series. We set the input length to 36, the predict length to 12, the patch length to 16, the encoder layer to 1, the model dimension to 64, and train with MSE loss for 10 epochs with learning rate of 0.001. For the general extractor model, we train an encoder-decoder

2. https://github.com/zhouhaoyi/ETDataset
3. http://pems.dot.ca.gov
4. https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
5. https://github.com/laiguokun/multivariate-time-series-data
6. https://www.bgc-jena.mpg.de/wetter/
7. https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

| Methods | Resource Type | ECL | ETTh1 | ETTh2 | Exchange | Illness | Traffic | Weather | Memory Storage (MB) |
|---|---|---|---|---|---|---|---|---|---|
| | | Downstream Target Dataset | | | | | | | Data + Model |
| Last | - | 0.7360 | 0.7640 | 0.2639 | **0.0217** | 4.7867 | 2.2498 | 1.4799 | - |
| Mean | | 0.6755 | 0.6314 | 0.3550 | 0.0376 | 4.8981 | 1.3565 | **1.4063** | |
| SeasonalNaive | | 0.6091 | 0.8539 | 0.3315 | 0.0272 | 6.0760 | 1.2227 | 1.6105 | |
| Arima | **In-Task Data** | 3.6648 | 0.6389 | 1.0048 | 10.1624 | 5.8628 | 2.4790 | 3.1264 | 0.01 + 30.27 |
| Prophet | | 10.2358 | 6.1366 | 10.1787 | 229.8594 | 9.1147 | 4.8610 | 2.9049 | 0.01 + 3.270 |
| Transformer | **In-Task Data** | 1.3429 | 0.6875 | 0.9453 | 1.5532 | 5.0552 | 1.9336 | 2.1727 | 0.01 + 64.06 |
| Autoformer | | 0.8861 | 0.8519 | 0.5835 | 0.1950 | 4.5547 | 1.4316 | 1.7660 | 0.01 + 65.88 |
| FEDformer | | 0.9156 | 0.7561 | 0.4718 | 0.0943 | 4.6087 | 1.5551 | 1.6792 | 0.01 + 80.38 |
| Informer | | 1.3743 | 0.7870 | 0.8497 | 1.5969 | 5.3082 | 2.0612 | 2.3070 | 0.01 + 67.07 |
| DLinear | | 0.6942 | 0.6472 | 0.3730 | 0.0559 | 4.8826 | 1.3655 | 1.4644 | 0.01 + 0.050 |
| PatchTST | | 0.6184 | 0.7333 | 0.4006 | 0.0354 | 3.9034 | 1.1661 | 1.4877 | 0.01 + 2.310 |
| iTransformer | | 0.6067 | 0.7183 | 0.3345 | 0.0315 | 3.5232 | **1.1306** | 1.5676 | 0.01 + 26.15 |
| Meta-N-BEATS | **Pre-Train Data** | 0.7576 | 0.7715 | 0.3133 | 0.0469 | 4.6405 | 2.2361 | 1.4648 | 1.70 + 95.85 |
| GPT4TS | | 0.7458 | 0.6961 | 0.3397 | 0.0280 | 6.9384 | 1.6730 | 1.4777 | 1.70 + 314.83 |
| ForecastPFN | | 0.9511 | 1.1851 | 0.5144 | 0.0579 | 4.8880 | 1.7894 | 1.8770 | * + 23.50 |
| SeqFusion | **PTMs** | **0.6029** | **0.6001** | **0.2450** | **0.0217** | **3.4956** | 1.4889 | 1.4488 | 0.02 + 23.10 |

TABLE 2
Performance comparisons on multivariate forecasting tasks over 5 trials, including classical statistical methods, deep learning models trained on 50 in-task time steps data, and zero-shot methods requiring pre-training data. SeqFusion achieves competitive performance across most datasets while requiring minimal memory storage. We denote the **best** and second-best results with bold-underline and underline. *Synthetic datasets.

| Methods | Resource Type | ECL | ETTh1 | ETTh2 | Exchange | Illness | Traffic | Weather | Memory Storage (MB) |
|---|---|---|---|---|---|---|---|---|---|
| | | Downstream Target Dataset | | | | | | | Data + Model |
| TimeGPT* | **Pre-Train Data** | 0.7458 | 0.6961 | 0.3397 | 0.0280 | 6.9384 | 1.6730 | 1.4777 | * |
| Chronos | | 0.5444 | 0.5652 | **0.2323** | **0.0239** | 5.7046 | 1.0351 | 1.3580 | $7.92\times10^5$ + 242.13 |
| Moirai | | 0.6909 | 0.6986 | 0.3102 | 0.0372 | 5.2899 | 1.4818 | 1.5127 | $3.56\times10^5$ + 365.15 |
| TimesFM | | 0.6100 | 0.5671 | 0.4721 | 0.0355 | **3.1143** | **0.8722** | 1.4332 | $\geq 1.00\times10^6$ + 814.76 |
| SeqFusion | **PTMs** | **0.5263** | **0.5265** | 0.2604 | 0.0244 | 4.3535 | 0.8775 | **1.3323** | 0.02 + $1.42\times10^3$ |

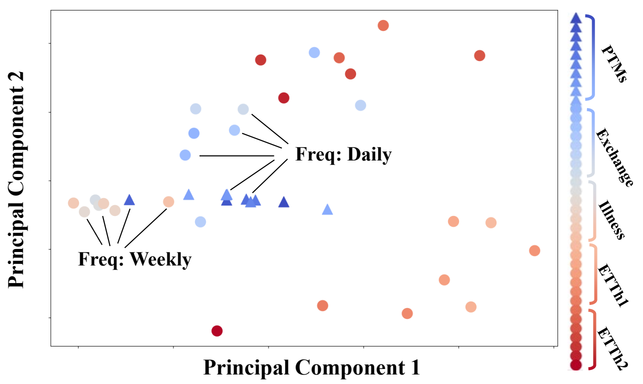
TABLE 3
Performance comparisons between large-scale pre-trained models and our SeqFusion framework, utilizing a large-scale pre-trained model zoo. SeqFusion achieves competitive or superior performance across most datasets with significantly lower memory storage requirements. We denote the **best** and second-best results with bold-underline and underline. *Closed-source.

model [42] as the general extractor based on the pre-training datasets of our PTMs to avoid data leakage. During training, we add a transferability loss to help representations extraction as mentioned above. Specifically, we randomly sample about 300,000 sub series from M3, M4 and Tourism. We conduct a cross-dataset evaluation to get the MSE performance of our model zoo, and use 1 minus this MSE performance (1- MSE) as the ground truth of the transferability loss. Since SeqFusion does not need training, once the model zoo is constructed, we can directly apply it for evaluation on target datasets, where we set the aggregated size to 3.

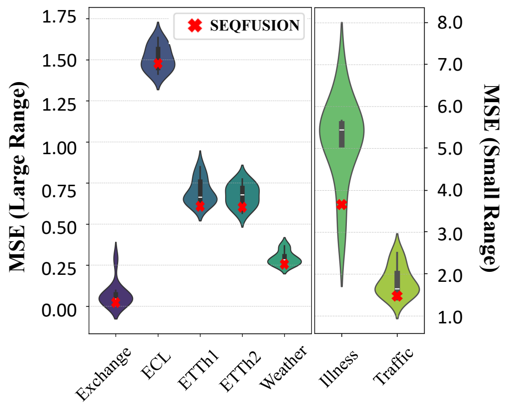
**Results**. Table 2 presents our findings for multivariate time-series forecasting tasks. We see that SeqFusion achieves the lowest and second lowest MSE value across all datasets compared to other baselines. For zero-shot methods, SeqFusion outperforms GPT4TS even though GPT4TS uses a GPT-2 model with a much larger number of parameters than our approach. We also find that the performance of naive baselines, Last, Mean, and SeasonalNaive depends on the dataset, being strong on some datasets and very weak at other times. Our method, on the other hand, shows consistent performance on all datasets.

SeqFusion is able to make good use of the PTM in the model zoo that best fits with the target time series for sequence prediction. As shown in Figure 4b, we plot the distribution of the ground true performance of the PTMs of our model zoo on all datasets, with the red "x" representing the combined performance of the selected PTMs with SeqFusion. It can be seen that in the vast majority of the datasets, SeqFusion selected a batch of the most suitable PTMs (box plot near the bottom). At the same time, the upper limit of the performance of SeqFusion is limited by the breadth of models in the model zoo. It can be found that even though SeqFusion selected the suitable PTMs for the traffic dataset, limited by the optimal PTMs, SeqFusion's performance is not satisfactory.

**Visualization**. To further illustrate why SeqFusion works, we perform some visualizations. First, we extract the representations of PTMs in the model zoo and the variate-wise representations of the target time series. We then apply PCA to reduce the dimensionality of these representations for visualization. As shown in Figure 4a, the triangles represent all PTMs and each circle represents one variate of the target time series. We first see that variates from the same dataset are cluster closer. Also, datasets with similar sampling frequencies (e.g., daily or weekly) are closer in the representation space to PTMs trained on datasets with matching frequencies. For instance, the Exchange dataset, which has a daily sampling frequency, is closely aligned with PTMs trained on the M4-Daily dataset. Similarly, datasets with weekly frequencies exhibit proximity to PTMs trained on datasets with similar temporal patterns. This visualization demonstrates that the representations effectively capture intrinsic characteristics of the time series by the general



(a) PCA plot of PTM repr. and target datasets repr.



(b) Performance distribution of all PTMs.

Fig. 4. (a) Visualization of the representations (repr.) of PTMs and target time series using PCA. Triangles represent PTMs, and circles represent variates from the target datasets, where we use similar colors to indicate different variate of the time series from one dataset. Variates from the same dataset cluster closely, and datasets with similar sampling frequencies (e.g., daily or weekly) are aligned with PTMs trained on datasets sharing these frequencies. (b) Violin plot of PTM performance distributions across datasets, with red "x" markers showing SEQFUSION 's combined performance. SEQFUSION consistently selects the most suitable PTMs in the model zoo (markers near the bottom), though its performance is limited by the quality and diversity of the model zoo.

extractor, such as sampling frequency, and align them with PTMs trained on datasets sharing these characteristics. Such alignment is critical for SEQFUSION 's success, as it ensures that the selected PTMs are not only relevant but also well-suited for the target time series.

## 5.2 Evaluation on Large Scale Pre-trained Models

**Setups** In the first experiment, we utilized a set of 10 lightweight PTMs, which demonstrates that even with minimal model complexity, our approach could achieve competitive results. To further evaluate SEQFUSION 's scalability and accuracy, we evaluate our method against state-of-the-art methods using pre-trained models, particularly those based on large language models or pre-trained on large-scale time-series datasets.

**Baselines**. We select several baselines represent various approaches to time-series forecasting using pre-trained models, which includes:

- **TimeGPT** [19]: a large-scale model pre-trained on diverse time series datasets. It demonstrates competitive or superior zero-shot forecasting performance when compared to fully supervised models. We use its official API to perform forecasting.
- **Chronos** [20]: a T5-based model that tokenizes time-series values using scaling and quantization, creating a fixed vocabulary for the data. Chronos employs transformer-based language model architectures, trained via cross-entropy loss, to process tokenized time-series sequences. We choose the official chronos-small version.
- **Moirai** [46]: a masked encoder-based universal time-series forecasting transformer, Moirai is trained on a large-scale datasets, which contains over 27 billion observations across nine domains. We choose the official Moirai-base version.
- **TimesFM** [47]: a large foundation model utilizing a patched-decoder attention mechanism, pre-trained on a large time-series corpus to capture temporal dependencies effectively.

**Implementation Details**. We construct 3 open-source pre-trained models to construct our model zoo and deploy the SEQFUSION framework. This involves Chronos [20], Moirai [46], and TimeFM [47].These models represent diverse architectures and pre-training approaches, ensuring that our model zoo captures a wide range of temporal patterns and forecasting capabilities. To demonstrate the generalization ability of SEQFUSION, we reuse the general extractor model trained in the first multivariate experiment, avoiding the need for task-specific re-training. To extract model representations, we sample 100,000 time-steps from the large-scale training datasets of Chronos, Moirai, and TimeFM as representative subsets. We assess the zero-shot performance on the multivariate forecasting tasks.

**Results** Table 3 demonstrates that SEQFUSION achieves competitive performance compared to state-of-the-art large-scale pre-trained models across diverse datasets while maintaining exceptional storage efficiency. For instance, SEQFUSION outperforms all baselines on ECL (**0.5263 MSE**) and Weather (**1.3323 MSE**), demonstrating its ability to handle datasets with complex temporal dependencies. Additionally, SEQFUSION provides robust accuracy on other datasets, such as Illness and ETTh2, showcasing its versatility across different forecasting domains.

A key advantage of SEQFUSION is its storage efficiency and its data privacy protection. While Moirai, Chronos, and TimesFM require storage of $3.56\times10^3$ MB, $7.92\times10^3$ MB, and over $1.00\times10^6$ MB, respectively, SEQFUSION only requires $1.42\times10^3$ MB. This reduction in memory requirements makes SEQFUSION a practical choice for resource-limited environments where large-scale models may be impractical to deploy. Moreover, SEQFUSION achieves this efficiency through collecting black-box PTMs with a few its pre-training data (less than 0.1%), making it particularly suitable for real-world applications that need to protect data privacy.

| Dataset | Granularity | Series Size | Look-back | Horizon | Mapping |
|---|---|---|---|---|---|
| M3 | Yearly | 645 | 12 | 6 | M4-Yearly |
| | Quarterly | 756 | 24 | 8 | M4-Quarterly |
| | Monthly | 1,428 | 24 | 18 | M4-Monthly |
| | Others | 174 | 16 | 8 | M4-Monthly |
| M4 | Yearly | 23,000 | 9 | 6 | M3-Yearly |
| | Quarterly | 24,000 | 16 | 8 | M3-Quarterly |
| | Monthly | 48,000 | 36 | 18 | M3-Monthly |
| | Weekly | 359 | 65 | 13 | M3-Monthly |
| | Daily | 4227 | 9 | 14 | M3-Monthly |
| Tourism | Yearly | 518 | 12 | 4 | M3-Yearly |
| | Quarterly | 427 | 24 | 8 | M3-Quarterly |
| | Monthly | 366 | 36 | 24 | M3-Monthly |

TABLE 4
Statistics of univariate time series datasets.

| Methods | Downstream Target Dataset | | |
|---|---|---|---|
| | M4 (SMAPE) | M3 (SMAPE) | Tourism (MAPE) |
| Last | 13.1685 | 16.2198 | 35.6551 |
| Mean | 18.1079 | 21.9992 | 47.3637 |
| SeasonalNaive | 16.5641 | 19.5313 | 40.6277 |
| Transformer | 152.6426 | 117.4006 | 134.8582 |
| Autoformer | 18.1650 | 17.6839 | 138.1960 |
| Informer | 152.6291 | 147.6470 | 133.9367 |
| DLinear | 15.3976 | 17.9384 | 34.9516 |
| PatchTST | 12.0447 | 13.7123 | **<u>25.4164</u>** |
| Meta-N-BEATS | 11.9463 | 13.3581 | <u>27.3988</u> |
| ForecastPFN | 18.1150 | 21.9990 | 47.3637 |
| GPT4TS | **<u>10.4272</u>** | **<u>11.6235</u>** | 33.7843 |
| SeqFusion | <u>11.1604</u> | <u>12.8145</u> | 30.4334 |

TABLE 5
Performance comparisons on univariate forecasting tasks over 5 trials. All methods except SeqFusion and naive baselines use training datasets with the mapped frequency of the target dataset.

The results also validate the scalability of SeqFusion, demonstrating its effectiveness across different model types, from lightweight to large-scale pre-trained models. By dynamically selecting and aggregating predictions from PTMs in the model zoo, SeqFusion achieves accuracy comparable to or better than large-scale pre-trained models. This flexibility ensures that SeqFusion is not only a powerful tool for zero-shot time-series forecasting but also a scalable and resource-efficient framework capable of adapting to a wide variety of forecasting scenarios.

## 5.3 Evaluation on Univariate Time Series

**Setups**. For a more comprehensive comparison of performance, we use the univariate time series benchmark from previous works [10], [18], including M3, M4 and Tourism. In this benchmark, we evaluate all baselines in a transfer setting, *i.e.*, how well a model trained from source dataset B performs on target dataset A (without any training data from A). For M4, Tourism forecasting tasks, we utilize M3 with the mapped granularity as the source dataset. For M3 forecasting tasks, we utilize M4 with the mapped granularity as the source dataset. Following [10], We use the source datasets with the same frequency as the target dataset in these baselines, although it is an unfair comparison to SeqFusion. We summarize the datasets in Table 4. We measure the performance using Mean Absolute Percentage Error (MAPE) and Symmetric Mean Absolute Percentage Error (SMAPE):

$$\mathrm{MAPE} = \frac{100}{H}\sum_{i=1}^{H}\frac{|\mathbf{Y}_i - \hat{\mathbf{Y}}_i|}{|\mathbf{Y}_i|} \tag{9}$$

$$\mathrm{SMAPE} = \frac{200}{H}\sum_{i=1}^{H}\frac{|\mathbf{Y}_i - \hat{\mathbf{Y}}_i|}{|\mathbf{Y}_i| + |\hat{\mathbf{Y}}_i|} \tag{10}$$

where we assume that $\mathbf{Y}, \hat{\mathbf{Y}} \in \mathbb{R}^{H\times 1}$ are the ground truth and prediction results of the future with $H$ time points and $C$ channels. $\mathbf{Y}_i$ denotes the $i$-th future time point.

**Implementation Details**. We construct the model zoo with 15 models of DLinear architecture pre-trained across traffic, weather, exchange, ETTh1 and ETTh2. We retrain a general extractor based on these pre-training data (randomly sample about 150,000 sub-series) to avoid data leakage from evaluation datasets. We set the aggregated size to 2.

**Results**. Table 5 presents the results for univariate time series. We observe that while GPT4TS with the powerful GPT2-backbone (around 314.83MB) achieves the highest rank, SeqFusion leveraging several simple MLP-based PTMs (only around 0.05MB) demonstrates a great alternative. This shows the importance of leveraging suitable models tailored to specific temporal patterns and forecasting tasks rather than using a general model (We will discuss this later in the ablation studies section). Unlike other deep learning baselines, SeqFusion does not require training and ranks second with no prior frequency information of the target time series, which highlights its potential for efficient time-series forecasting across various applications in a zero-shot setting.

## 5.4 Ablation Studies

We analyze the properties of SeqFusion following the multivariate benchmark.

**General or Specialized**. The model zoo we constructed consists of PTMs well-trained on diverse datasets, each acquiring *specialized* knowledge. When addressing downstream tasks, SeqFusion is designed to select the most suitable PTMs for forecasting. To illustrate the effectiveness and necessity of this selection process, we trained a *general* model using PatchTST on the entire pre-training data of our model zoo. Table 6 demonstrates that the generally pre-trained model performs worse than the selected specialized PTMs. This underscores the importance of leveraging specialized models tailored to specific temporal patterns and forecasting tasks. By selecting PTMs that are finely tuned to particular characteristics of the target time series, SeqFusion can achieve more accurate and reliable predictions. This approach not only enhances forecasting performance but also mitigates the need for extensive and potentially privacy-compromising data collection. Furthermore, the ability to dynamically choose the most appropriate models from the zoo allows SeqFusion to adapt to a wide range of forecasting scenarios, making it a versatile and robust solution.

**Model Zoo Family**. We conduct experiments to investigate how the composition and variety of the model zoo impact the performance of SeqFusion. Firstly, we examine the effect of varying the number of PTMs in the model zoo. We increase the number of PTMs from the original 10 to 20,

| Methods | Downstream Target Dataset | | | | | | |
|---|---|---|---|---|---|---|---|
| | ECL | ETTh1 | ETTh2 | Exchange | illness | traffic | weather |
| General Model | 0.8291 | 0.7723 | 0.3173 | 0.0352 | 5.7027 | 1.6742 | 1.5435 |
| Specialized Models | **0.6029** | **0.6001** | **0.2450** | **0.0217** | **3.4956** | **1.4889** | **1.4488** |

TABLE 6
Performance comparisons of SEQFUSION with a general model pre-trained on the entire pre-training data of model zoo. Specialized PTMs selected by SEQFUSION outperform the general model across all tasks. This highlights the importance of leveraging specialized models, as SEQFUSION dynamically selects PTMs tailored to the temporal patterns and forecasting requirements of the target time series, making full use of PTM resource and reducing extensive data collection.

including 3 PTMs trained on the newly added Hospital [29] dataset and 7 PTMs trained on data sampled from the M3 and M4 datasets. We also explore the impact of different PTM architectures within the model zoo. Specifically, we mixed 10 PatchTST architecture (transformer-based) with 10 DLinear architecture (MLP-based). The final results are shown in the Table 7.

| Model Zoo | Datasets | | | |
|---|---|---|---|---|
| | ECL | ETTh1 | Illness | Weather |
| PatchTST x 10 | **0.6087** | **0.6001** | 1.4488 | 3.6602 |
| + DLinear x 10 | 0.6156 | **0.6001** | 1.4488 | 5.1615 |
| PatchTST x 20 | 0.6281 | 0.6395 | **1.4471** | **3.3798** |

TABLE 7
MSE comparisons for different model zoo configurations. Increasing PTMs from 10 to 20, with models from Hospital, M3, and M4 datasets, improves performance on some datasets. Mixing architectures (PatchTST + DLinear) reduces accuracy.

Increasing the number of PTMs from 10 to 20 shows a marginal improvement in performance on certain datasets, particularly the Illness dataset, where the MSE is slightly reduced from 1.4488 to 1.4471. This improvement can be attributed to the inclusion of PTMs trained on the Hospital dataset, which shares similar temporal and domain-specific characteristics with the Illness dataset. The knowledge transfer from related datasets (in this case, medical datasets with similar recording frequencies) appears to enhance SeqFusion's ability to forecast accurately within this domain. In contrast, adding architectural diversity by mixing PatchTST (transformer-based) and DLinear (MLP-based) models leads to a decrease in performance, especially noticeable on datasets like Weather, where the MSE increases from 3.6602 to 5.1615. This suggests that while diversity in model architecture might introduce new perspectives, it can also disrupt the coherence of the model zoo, potentially introducing conflicting patterns or biases that reduce the ensemble's effectiveness. Transformer-based architectures like PatchTST appear more suited to capturing temporal dependencies in these time-series datasets compared to MLP-based models like DLinear. Overall, these results indicate that to maximize SEQFUSION 's performance, it is essential to expand the model zoo with PTMs that capture diverse temporal patterns from relevant domains, rather than merely increasing architectural variety. In other words, incorporating PTMs trained on domain-similar datasets (e.g., Hospital for Illness) is more beneficial than mixing fundamentally different architectures, which may not align well with the specific temporal characteristics of the target datasets.

**How does transferability loss help?** To illustrate the effect of transferability loss in the training of the general extractor, we first compare the performance of SEQFUSION using general extractor models with/without transferability loss. In Table 8, it shows that the general extractor with transferability loss outperforms the one without consistently. In fact, the transferability loss helps PTMs trained from related datasets to be similar in the representation space. We use PCA to visualize all PTMs' representations extract from the general extractor model trained with the transferability loss. In Figure 4a, we can see that PTMs trained from the same source dataset are closer after using the transferability loss and vice versa. We also consider replace the architecture of the general extractor model with another time-series representation method TS2Vec [41]. In Table 8, we observe that TS2Vec demonstrates comparable performance to SimMTM. However, SimMTM with transferability loss shows a slight advantage on some more challenging tasks, such as ECL and weather dataset. This performance boost suggests that transferability loss helps align representations from related datasets, enhancing the model's ability to generalize across domains, which appears critical for tasks requiring cross-domain generalization.

**Enhancing forecasting with PTM Aggregation.** SEQFUSION is able to incorporate top suitable PTMs to improve forecasting. Figure 5 demonstrates that SEQFUSION benefits from aggregating predictions from multiple PTMs, with MSE decreasing as the number of aggregated PTMs increases. By combining predictions from multiple PTMs, SeqFusion effectively reduces the impact of any single model's potential biases or errors, leading to more reliable final predictions. This ensemble approach allows SeqFusion to leverage the strengths of individual PTMs while mitigating weaknesses, resulting in an overall improvement in performance. The declining trend in average MSE as the number of PTMs increases suggests that aggregating predictions is a robust strategy for enhancing predictive accuracy across varied time-series forecasting tasks.

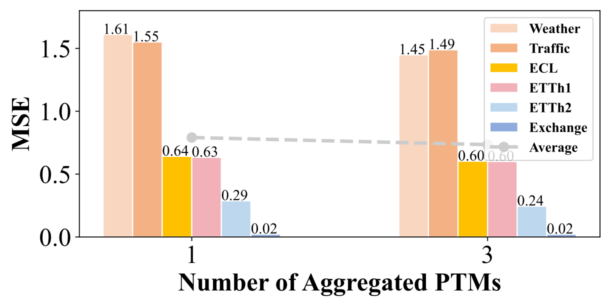


Fig. 5. Performance of SEQFUSION with varying numbers of aggregated PTMs. Aggregating predictions from multiple PTMs reduces MSE across datasets.

| Methods | Downstream Target Dataset | | | | | | |
|---|---|---|---|---|---|---|---|
| | ECL | ETTh1 | ETTh2 | Exchange | illness | traffic | weather |
| TS2Vec [41] | 0.6061 | **0.5723** | 0.2673 | 0.0247 | **3.3150** | **1.3748** | 1.4675 |
| SimMTM [42] w/o Trans. | 0.6056 | 0.6164 | 0.2490 | 0.0239 | 3.4983 | 1.4360 | 1.4548 |
| SimMTM [42] w/ Trans. | **0.6029** | 0.6001 | **0.2450** | **0.0217** | 3.4956 | 1.4889 | **1.4488** |

TABLE 8
Performance comparisons of SEQFUSION using general extractor models with TS2Vec and SimMTM architectures (with/without transferability loss). While TS2Vec achieves comparable performance to SimMTM, the addition of transferability loss enhances performance across datasets.

## 6 CONCLUSION

SEQFUSION introduces a novel framework for zero-shot time-series forecasting that leverages the power of a diverse collection of pre-trained models (PTMs). Unlike traditional methods that require extensive in-task training data or generalized pre-trained models dependent on large-scale datasets, SEQFUSION dynamically selects and aggregates predictions from PTMs specifically tailored to the characteristics of the target time series. This approach enhances privacy, reduces storage costs, and increases model reuse flexibility. Through recursive forecasting and aggregated predictions, SEQFUSION achieves competitive accuracy across various datasets, demonstrating robustness and adaptability in data-limited environments. Experiments demonstrate SEQFUSION's potential to outperform state-of-the-art methods, showcasing its flexibility and practicality in diverse domains such as healthcare, finance, and environmental science. A key innovation in SEQFUSION is its ability to extract these representations using only a small amount of data that does not compromise privacy. For example, in financial applications like stock market forecasting, where proprietary trading data must remain confidential, SEQFUSION can rely on limited, non-sensitive indicators such as sector-level averages or public historical stock prices to extract meaningful representations. These minimal yet informative datasets allow the feature extractor to align target time series with suitable PTMs without requiring access to detailed or proprietary raw data.

The success of SEQFUSION highlights the potential for PTM-based approaches in time-series forecasting. Future work could explore more sophisticated model selection and fusion strategies, such as adaptive weighting or task-specific optimization. Additionally, integrating more PTMs from diverse domains and the extension of SEQFUSION to handle real-time forecasting tasks could further enhance its applicability. By demonstrating the feasibility of sharing and reusing PTMs for time-series applications, SEQFUSION paves the way for advancing zero-shot forecasting methodologies and their adoption in industry and research.

## ACKNOWLEDGMENTS

This work is partially supported by the National Key R&D Program of China (2022ZD0114805), NSFC (62376118, 62176117, 62272231), NSF of Jiangsu Province (BK20200313, BK20210340), Collaborative Innovation Center of Novel Software Technology and Industrialization, the Fundamental Research Funds for the Central Universities (No. NJ2022028).

## REFERENCES

[1] R. B. Penfold and F. Zhang, "Use of interrupted time series analysis in evaluating health care quality improvements," *Academic pediatrics*, vol. 13, no. 6, pp. S38–S44, 2013.
[2] O. B. Sezer, M. U. Gudelek, and A. M. Özbayoglu, "Financial time series forecasting with deep learning : A systematic literature review: 2005-2019," *Appl. Soft Comput.*, vol. 90, p. 106181, 2020.
[3] Z. Karevan and J. A. K. Suykens, "Transductive LSTM for time-series prediction: An application to weather forecasting," *Neural Networks*, vol. 125, pp. 1–9, 2020.
[4] R. G. Hoptroff, "The principles and practice of time series forecasting and business modelling using neural nets," *Neural Comput. Appl.*, vol. 1, no. 1, pp. 59–66, 1993.
[5] G. E. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.
[6] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *AAAI*, 2021, pp. 11 106–11 115.
[7] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
[8] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *ICLR*, 2023.
[9] L. Han, H. Ye, and D. Zhan, "The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 7129–7142, 2024.
[10] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One fits all: Power general time series analysis by pretrained LM," in *NeurIPS*, 2023.
[11] L. Han, X. Chen, H. Ye, and D. Zhan, "SOFTS: efficient multivariate time series forecasting with series-core fusion," *CoRR*, vol. abs/2404.14197, 2024.
[12] Z. Li, R. Cai, T. Z. J. Fu, Z. Hao, and K. Zhang, "Transferable time-series forecasting under causal conditional shift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 4, pp. 1932–1949, 2024. [Online]. Available: https://doi.org/10.1109/TPAMI.2023.3304354
[13] G. Spadon, S. Hong, B. Brandoli, S. Matwin, J. F. R. Jr., and J. Sun, "Pay attention to evolution: Time series forecasting with deep graph-evolution learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5368–5384, 2022.
[14] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120 043–120 065, 2021.
[15] R. M. C. Croda, D. E. G. Romero, and S. O. C. Morales, "Sales prediction through neural networks for a small dataset," *Int. J. Interact. Multim. Artif. Intell.*, vol. 5, no. 4, pp. 35–41, 2019.
[16] V. Cerqueira, L. Torgo, and C. Soares, "Machine learning vs statistical methods for time series forecasting: Size matters," *CoRR*, vol. abs/1909.13316, 2019.
[17] S. J. Fong, G. Li, N. Dey, R. G. Crespo, and E. Herrera-Viedma, "Finding an accurate early forecasting model from small dataset: A case of 2019-ncov novel coronavirus outbreak," *Int. J. Interact. Multim. Artif. Intell.*, vol. 6, no. 1, pp. 132–140, 2020.
[18] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "Meta-learning framework with applications to zero-shot time-series forecasting," in *AAAI*, 2021, pp. 9242–9250.
[19] A. Garza and M. M. Canseco, "Timegpt-1," *CoRR*, vol. abs/2310.03589, 2023.

[20] A. F. Ansari, L. Stella, A. C. Türkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. Pineda-Arango, S. Kapoor, J. Zschiegner, D. C. Maddix, M. W. Mahoney, K. Torkkola, A. G. Wilson, M. Bohlke-Schneider, and Y. Wang, "Chronos: Learning the language of time series," *CoRR*, vol. abs/2403.07815, 2024.

[21] G. Kaissis, M. R. Makowski, D. Rueckert, and R. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nat. Mach. Intell.*, vol. 2, no. 6, pp. 305–311, 2020.

[22] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, and Z. Lin, "When machine learning meets privacy: A survey and outlook," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 31:1–31:36, 2022.

[23] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electron. Mag.*, vol. 9, no. 3, pp. 8–16, 2020.

[24] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *CoRR*, vol. abs/1910.03771, 2019.

[25] Y. Li and Y. Pan, "A novel ensemble deep learning model for stock prediction based on stock prices and news," *Int. J. Data Sci. Anal.*, vol. 13, no. 2, pp. 139–149, 2022.

[26] Y. Zhang, T. Huang, Y. Ding, D. Zhan, and H. Ye, "Model spider: Learning to rank pre-trained models efficiently," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[27] M. Li, Y. Zhu, Y. Shen, and M. Angelova, "Clustering-enhanced stock price prediction using deep learning," *World Wide Web (WWW)*, vol. 26, no. 1, pp. 207–232, 2023.

[28] Z. Zhou and Z. Tan, "Learnware: small models do big," *Sci. China Inf. Sci.*, vol. 67, no. 1, 2024.

[29] R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[30] M. Jin, H. Y. Koh, Q. Wen, D. Zambon, C. Alippi, G. I. Webb, I. King, and S. Pan, "A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 10 466–10 485, 2024.

[31] O. Styles, T. Guha, and V. Sanchez, "Multi-camera trajectory forecasting with trajectory tensors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8482–8491, 2022. [Online]. Available: https://doi.org/10.1109/TPAMI.2021.3107958

[32] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *NeurIPS*, 2021, pp. 101–112.

[33] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *ICML*, vol. 162, 2022, pp. 27 268–27 286.

[34] S. Dooley, G. S. Khurana, C. Mohapatra, S. V. Naidu, and C. White, "Forecastpfn: Synthetically-trained zero-shot forecasting," in *NeurIPS*, 2023.

[35] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson, "Large language models are zero-shot time series forecasters," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[36] X. Zhan, J. Xie, Z. Liu, Y. Ong, and C. C. Loy, "Online deep clustering for unsupervised representation learning," in *CVPR*, 2020, pp. 6687–6696.

[37] Q. Ma, J. Zheng, S. Li, and G. W. Cottrell, "Learning representations for time series clustering," in *NeurIPS*, 2019, pp. 3776–3786.

[38] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pre-trained deep recurrent neural network for time series classification," in *ESANN*, 2017.

[39] Q. Lei, J. Yi, R. Vaculín, L. Wu, and I. S. Dhillon, "Similarity preserving representation learning for time series clustering," in *IJCAI*, 2019, pp. 2845–2851.

[40] S. Tonekaboni, D. Eytan, and A. Goldenberg, "Unsupervised representation learning for time series with temporal neighborhood coding," in *ICLR*, 2021.

[41] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "Ts2vec: Towards universal representation of time series," in *AAAI*, 2022, pp. 8980–8987.

[42] J. Dong, H. Wu, H. Zhang, L. Zhang, J. Wang, and M. Long, "Simmtm: A simple pre-training framework for masked time-series modeling," in *NeurIPS*, 2023.

[43] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *ICLR*, 2019.

[44] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski, "MOMENT: A family of open time-series foundation models," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[45] V. Ekambaram, A. Jati, N. H. Nguyen, P. Dayama, C. Reddy, W. M. Gifford, and J. Kalagnanam, "Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series," *CoRR*, vol. abs/2401.03955, 2024.

[46] G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo, "Unified training of universal time series forecasting transformers," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[47] A. Das, W. Kong, R. Sen, and Y. Zhou, "A decoder-only foundation model for time-series forecasting," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[48] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *AAAI*, B. Williams, Y. Chen, and J. Neville, Eds., 2023, pp. 11 121–11 128.

[49] S. B. Taieb, G. Bontempi, A. Sorjamaa, and A. Lendasse, "Long-term prediction of time series by combining direct and MIMO strategies," in *International Joint Conference on Neural Networks, IJCNN 2009, Atlanta, Georgia, USA, 14-19 June 2009*. IEEE Computer Society, 2009, pp. 3054–3061.

[50] S. B. Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 7067–7083, 2012. [Online]. Available: https://doi.org/10.1016/j.eswa.2012.01.039

[51] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *SIGIR*, 2018, pp. 95–104.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.