BLAST: Balanced Sampling Time Series Corpus for Universal Forecasting Models

Zezhi Shao*
Institute of Computing Technology,
Chinese Academy of Sciences
State Key Laboratory of AI Safety
University of Chinese Academy of
Sciences
shaozezhi@ict.ac.cn

Yujie Li*
Institute of Computing Technology,
Chinese Academy of Sciences
State Key Laboratory of AI Safety
University of Chinese Academy of
Sciences
liyujie23s@ict.ac.cn

Fei Wang†
Institute of Computing Technology,
Chinese Academy of Sciences
State Key Laboratory of AI Safety
University of Chinese Academy of
Sciences
wangfei@ict.ac.cn

Chengqing Yu, Yisong Fu
Institute of Computing Technology,
Chinese Academy of Sciences
State Key Laboratory of AI Safety
University of Chinese Academy of
Sciences
{yuchengqing22b,fuyisong24s}@ict.ac.cn

Tangwen Qian, Bin Xu, Boyu
Diao
Institute of Computing Technology,
Chinese Academy of Sciences
State Key Laboratory of AI Safety
University of Chinese Academy of
Sciences
{qiantangwen,xubin,diaoboyu2012}@ict.ac.cn

Yongjun Xu, Xueqi Cheng
Institute of Computing Technology,
Chinese Academy of Sciences
State Key Laboratory of AI Safety
University of Chinese Academy of
Sciences
{xyj,cxq}@ict.ac.cn

Abstract

The advent of universal time series forecasting models has revolutionized zero-shot forecasting across diverse domains, yet the critical role of data diversity in training these models remains underexplored. Existing large-scale time series datasets often suffer from inherent biases and imbalanced distributions, leading to suboptimal model performance and generalization. To address this gap, we introduce BLAST, a novel pre-training corpus designed to enhance data diversity through a balanced sampling strategy. First, BLAST incorporates 321 billion observations from publicly available datasets and employs a comprehensive suite of statistical metrics to characterize time series patterns. Then, to facilitate pattern-oriented sampling, the data is implicitly clustered using grid-based partitioning. Furthermore, by integrating grid sampling and grid mixup techniques, BLAST ensures a balanced and representative coverage of diverse patterns. Experimental results demonstrate that models pre-trained on BLAST achieve state-of-the-art performance with a fraction of the computational resources and training tokens required by existing methods. Our findings highlight the pivotal role of data diversity in improving both training efficiency and model performance for the universal forecasting task.

CCS Concepts

• **Information systems → Data mining**.

---

*Equal contribution.
†Corresponding author.

Keywords

large-scale time series dataset, balanced sampling, universal time series forecasting

KDD Availability Link:
The code for training universal forecasting models with BLAST is available at https://github.com/GestaltCogTeam/BasicTS, and the BLAST generation code can be found at https://github.com/GestaltCogTeam/BLAST.

1 Introduction

Universal time series forecasting models have introduced new possibilities for accurate zero-shot forecasting across various domains [3, 9, 15, 24, 34, 37, 43]. One of the most critical foundations for training these models lies in large-scale and diverse datasets. Consequently, acquiring and organizing these training corpora has emerged as a crucial challenge.

A large-scale time series dataset is typically composed of multiple sub-datasets, where candidate samples are generated using a sliding window on each sequence and subsequently sampled to obtain data for model training. An example of a large-scale dataset consisting of three sub-datasets is illustrated in Figure 1(a). It is worth noting that the sequence length and the number of sequences may vary significantly across sub-datasets. Recent pioneering studies [3, 12, 24, 37, 43] have leveraged multi-domain data to construct large-scale time series datasets. For instance, the LOTSA [43] dataset contains over 231 billion observations (considering all variates), while the Time-300B [37] dataset is even larger, with 309 billion

(a) A large-scale raw dataset $\mathcal{D}$, composed of three sub-datasets $\{\mathcal{D}_1, \mathcal{D}_2,$ and $\mathcal{D}_3\}$.

(b) Candidate sample set $\mathcal{W}$. (d) Stratified sampling.
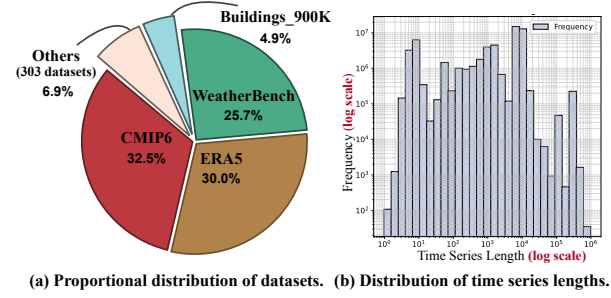
(c) Naive sampling.

Figure 1: Illustration of the large-scale time series forecasting pre-training dataset and various sampling methods.

observations. These studies primarily focus on the *scale* of data, laying a solid foundation for training universal forecasting models.

Despite the growing scale, the *diversity* of pre-training data has not yet been investigated. High-quality training data should capture a wide array of patterns while ensuring balanced sample sizes for each [1, 29, 35]. However, the initial distribution of large-scale time series datasets is often highly imbalanced. As illustrated in Figure 2(a), only three datasets account for 88.2% of the total data volume, and Figure 2(b) further highlights the imbalance in sequence lengths, where longer sequences tend to contribute disproportionately more samples. These skewed distributions will result in numerous repetitive patterns [37, 41] in the raw data, compromising overall data diversity. Thus, how to sample data with rich and balanced patterns becomes a crucial challenge.

However, existing studies generally overlook these imbalance issues, adopting simplistic sampling strategies such as naive sampling or stratified sampling. The former uniformly selects samples from all sub-datasets, as shown in Figure 1(c). The latter usually involves two steps: first, uniformly or weightedly selecting a sub-dataset (or sub-domain), and then selecting a sample within that sub-dataset, as illustrated in Figure 1(d). While these sampling strategies are intuitive and easy to implement, they fail to sufficiently correct for the inherent biases in large-scale time series data. Specifically, while naive sampling entirely overlooks these biases, stratified sampling attempts to mitigate them, but often assumes that data within the same dataset or domain share similar patterns, which is reasonable but does not always hold. For instance, as shown in Figure 1(a), both $\mathcal{D}_1$ and $\mathcal{D}_3$ originate from the traffic domain but exhibit distinct patterns. Similarly, two time series within $\mathcal{D}_2$ display divergent patterns. In summary, the inability to ensure diversity in the training data can have significant negative consequences. For example, the model may overfit to frequent patterns while underfitting less common ones, impairing its generalization capability.

To address the aforementioned issues, we propose a novel pre-training corpus named BLAST (BaLAnced Sampling Time series corpus). First, we integrate a wide range of publicly available datasets, creating a large-scale dataset with a total of 321 billion observations. Unlike prior approaches that depend on dataset or domain labels to differentiate time series patterns, BLAST incorporates a diverse array of statistical attributes to comprehensively characterize each

(a) Proportional distribution of datasets. (b) Distribution of time series lengths.

Figure 2: The uneven distribution of the raw large-scale time series dataset collected by BLAST.

time series' patterns, such as stationarity, seasonality, volatility, *etc.* Subsequently, BLAST amalgamates these heterogeneous features into unified feature vectors through a discretization process and projects them into a low-dimensional space, thereby intuitively revealing the uneven distribution of the data. Then, BLAST employs grid sampling and grid mixup within the low-dimensional space to ensure a balanced and representative coverage of diverse patterns.

To validate the effectiveness of BLAST, we trained state-of-the-art universal forecasting models using the proposed corpus *from scratch*. Table 1 presents the results from the TimeMoE [37] model.[1] The original TimeMoE was trained on *419 billion tokens* using *128 A100 GPUs*. In contrast, the BLAST-based TimeMoE achieves state-of-the-art performance with only *78 billion tokens* and *8 A100 GPUs*. These results demonstrate that incorporating data diversity allows BLAST-based model training to achieve substantial advantages in both training efficiency and model performance.

In summary, the key contributions are as follows:

- This study fills a critical gap in the role of data diversity in training universal forecasting models. It is the first to investigate the effect of pre-training data diversity on training efficiency and model performance.
- We propose a balanced sampling technique that treats time series patterns as the sampling target. Specifically, the time series is characterized by multiple statistical properties, and data is implicitly clustered using grid-based partitioning. Grid sampling and grid mixup techniques are then applied to generate diversified pre-training data.
- We develop BLAST, an efficient time series corpus generated through the balanced sampling. Experimental results show that BLAST-based pre-training achieves superior performance while reducing resource and data requirements.

2 Preliminaries

In this section, we define the notions of large-scale time series forecasting datasets, sampling strategies, and universal forecasting models. Frequently used notations are summarized in Table 2.

DEFINITION 1. *Large-scale Time Series Forecasting Dataset* $\mathcal{D}$ *comprises* N *sub-datasets, denoted as* $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N$. *Each sub-dataset* $\mathcal{D}_n$ *contains* K_n *time series* $\{X_1^n, X_2^n, \ldots, X_{K_n}^n\}$. *The k-th time*

---

[1]The choice of TimeMoE as the baseline for presenting the results is motivated by two main reasons: (i) TimeMoE is the only model pre-trained on a comparably large-scale raw dataset (309 billion observations); and (ii) TimeMoE is recognized as one of the state-of-the-art universal forecasting models.

Table 1: Comparison of training cost and performance between TimeMoE$_{base}$-BLAST and the original TimeMoE$_{base}$. The average MSE/MAE is reported as shown in Table 5.

	TimeMoE$_{base}$-BLAST	TimeMoE$_{base}$
Hardware	8×A100	128×A100
# Batch Size	192	1024
# Training Tokens	78.64B	419.43B
Avg. MSE / MAE	0.325 / 0.368	0.341 / 0.385

series X_k^n in the n-th sub-dataset consists of T_{nk} time steps, denoted as $X_k^n = \{x_1^{nk}, x_2^{nk}, \ldots, x_{T_{nk}}^{nk}\}$. Note that the size of the sub-datasets and the length of individual time series can vary significantly.

DEFINITION 2. **Sampling Strategies** *refer to the methods used to select training data from candidate sample set $\mathcal{W}$. Raw time series cannot be directly used for model training. Candidate samples $\mathcal{W}$ are generated by applying a sliding window W to each time series. The goal of the sampling strategy is to select the final set of samples used for training from these candidates.*

DEFINITION 3. **Universal Forecasting Models**[2] *are pre-trained on large-scale time series datasets and are capable of performing accurate zero-shot forecasting across diverse domains.*

3 Related Work

3.1 Universal Time Series Forecasting

Inspired by breakthroughs in artificial intelligence [18, 31, 36, 38], universal time series forecasting aim to achieve zero-shot forecasting across domains through pre-training on large-scale datasets.

These models are predominantly built on Transformer architectures [40] and can be categorized into encoder-only models [12, 14, 17, 43], decoder-only models [9, 23–25, 34, 37], and encoder-decoder models [3, 15]. *Encoder-only* models typically employ masked encoding strategies along with architectures tailored for time series tasks. *Decoder-only* models, on the other hand, often utilize autoregressive pre-training strategies. Recent advancements have incorporated techniques such as mixture-of-experts [21, 37], long-context modeling [23], and hierarchical modeling approaches [25] to further improve their capabilities. *Encoder-decoder* [3, 15] architectures retain the full Transformer framework for time series tasks. In parallel, cutting-edge research [8, 13, 42] have begun exploring architectures beyond Transformers or other modalities [6, 19, 20], aiming to design models specifically for time series data and further enhance forecasting accuracy.

Overall, these universal models demonstrate surprising zero-shot forecasting capabilities through pre-training on large-scale datasets, underscoring their transformative potential in this field.

3.2 Time Series Forecasting Pre-training Corpus

Regardless of the model architectures, large-scale pre-training data $\mathcal{D}$ serves as the foundation for achieving universal forecasting. The

---

[2]While some studies refer to these models as foundational or general models, this paper adopts the term *universal forecasting models* [2, 14, 42, 43] for the sake of consistency and to avoid confusion with multi-task models.

Table 2: Frequently used notations.

Notations	Definitions
$\mathcal{D}$	$\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N\}$ is the raw large-scale pre-training dataset, consisting of N sub-datasets.
$\mathcal{D}_n$	$\mathcal{D}_n = \{X_1^n, X_2^n, \ldots, X_{K_n}^n\}$ is the n-th sub-dataset, containing K_n time series.
X_k^n	$X_k^n = \{x_1^{nk}, x_2^{nk}, \ldots, x_{T_{nk}}^{nk}\}$ is the k-th time series in the n-th sub-dataset $\mathcal{D}_n$, containing T_{nk} time steps.
x_t^{nk}	x_t^{nk} is the t-th time step in the time series X_k^n.
$\mathcal{W}$	$\mathcal{W}$ denotes the collection of context windows drawn from $\mathcal{D}$.
W	W denotes the data under a context window of length $|W|$.
S	S denotes the stride of the sliding context window; throughout this paper we set $S = 1$ by default.
$\lfloor \cdot \rfloor$	$\lfloor \cdot \rfloor$ denotes the floor operation.

size of the raw data is typically measured by the total number of observations, expressed as $\sum_{n=1}^{N} \sum_{k=1}^{K_n} T_{nk}$.

Numerous pioneering works have established large-scale training corpora to support universal forecasting models. For instance, ForecastPFN [12] innovatively explored the role of purely synthetic data in pre-training. Chronos [3] combined data from sources such as Monash [16] and M-competitions [27], as well as synthetic data, to create a corpus with a total of 84 billion observations. Similarly, MOIRAI [43] introduced the large-scale dataset, LOTSA, which includes 231 billion observations (accounting for all variates). Timer [24] developed the UTSD dataset by collecting multi-domain data, comprising 1 billion observations. Another example is TimeMoE [37], which constructed the largest existing dataset, Time-300B, by integrating various data sources, reaching a scale of 309 billion observations. These contributions have laid a solid foundation for the development of universal forecasting models.

While most of these studies have focused primarily on the *scale* of data, systematic investigations into *diversity* remain unexplored. To address this gap, we propose a diversified pre-training corpus—BLAST. Built on 321 billion raw observations, BLAST leverages a balanced sampling strategy to ensure diversity. We select state-of-the-art models and retrain them on the BLAST corpus. Experimental results demonstrate that pre-training on BLAST is superior significantly in both training efficiency and model performance, underscoring the importance of a diversified corpus.

4 Limitations of Existing Sampling Strategies

The purpose of a sampling strategy is to select training samples from the candidate sample set $\mathcal{W}$. This set is generated by applying a sliding window with stride S to each time series. Each sample, denoted as $W_{n,k,t}$, corresponds the t-th sliding window position of the k-th time series in the n-th sub-dataset. The sampling strategy is defined by the probability distribution $\mathbb{P}(W_{n,k,t})$.

4.1 Naive Sampling

The most straightforward way is the naive sampling, which uniformly selects the candidate samples:

$$\mathbb{P}(W_{n,k,t}) = \text{Uniform}(\mathcal{W}) = \frac{1}{|\mathcal{W}|}. \tag{1}$$

Figure 3: Pipeline for the balanced sampling: (i) constructing large-scale time series datasets, (ii) utilizing diverse metrics to comprehensively characterize time series, (iii) generating unified feature vectors and performing dimension reduction to visualize data imbalances, and (iv) implementing grid sampling and grid mixup to enhance the diversity of the training data.

$\mathcal{W}$ is the candidate sample set, and is formally defined as:

$$\mathcal{W} = \bigcup_{n=1}^{N} \bigcup_{k=1}^{K_n} \{W_{n,k,t} \mid t = 1, 1+S, \ldots, \text{and } t + |W| - 1 \le T_{nk}\},$$

$$|\mathcal{W}| = \sum_{n=1}^{N} \sum_{k=1}^{K_n} \left\lfloor \frac{T_{nk} - |W|}{S} \right\rfloor + 1. \quad (2)$$

These notations are defined in Section 2 and Table 2.

4.2 Stratified Sampling

Stratified sampling typically involves selecting a sub-dataset (uniformly or with weighted probabilities) and then applying naive sampling within it. The stratified sampling (uniform) can be defined as:

$$\mathbb{P}(W_{n,k,t}) = \mathbb{P}(\mathcal{W}_n) \cdot \mathbb{P}(W_{n,k,t} \mid \mathcal{W}_n),$$

$$\mathbb{P}(\mathcal{W}_n) = \frac{1}{N}, \quad \mathbb{P}(W_{n,k,t} \mid \mathcal{W}_n) = \frac{1}{|\mathcal{W}_n|}, \quad (3)$$

where N is the number of sub-datasets, $\mathcal{W}_n$ is the candidate sample set generate from sub-dataset $\mathcal{D}_n$, and can be defined as:

$$\mathcal{W}_n = \bigcup_{k=1}^{K_n} \{W_{n,k,t} \mid t = 1, 1+S, \ldots, \text{and } t + |W| - 1 \le T_{nk}\},$$

$$|\mathcal{W}_n| = \sum_{k=1}^{K_n} \left\lfloor \frac{T_{nk} - |W|}{S} \right\rfloor + 1. \quad (4)$$

Table 3: Comparison of time series corpora.

Corpus	Raw Size	Open Source	Sampling Strategy
UTSD [24]	1B	✓	Naive Sampling
MOMENT [17]	1.23B	✓	Naive Sampling
Chronos [3]	84B	✓	Stratified Sampling
TimeGPT [15]	~100B	✗	Unknown
LOTSA [43]	231B	✓	Stratified Sampling
TimesFM [9]	~307B	✗	Unknown
Time-300B [37]	309B	✓	Naive Sampling
BLAST	**321B**	**✓**	**Balanced Sampling**

4.3 The Limitations

An effective sampling strategy should generate samples with rich pattern while maintaining balanced sample sizes across patterns, *i.e.*, diversity. However, naive sampling preserves the original data structure and its inherent biases. Stratified sampling partially addresses this issue, but the assumption that domain or dataset labels reliably differentiate time series patterns is flawed. Table 3 summarizes the corpus and sampling strategies in existing studies, most of which rely on naive sampling or stratified sampling, or their improved variants. For instance, MOIRAI [43] proposes the LOTSA dataset and employs a weighted stratified sampling approach with thresholds. In summary, these simple strategies often lead to uneven data distributions, negatively affecting the model's convergence and generalization ability.

5 Balanced Sampling Time Series Corpus

The core insight of BLAST lies in harnessing the diverse statistical characteristics of time series data to *implicitly* cluster the data through grid-based partitioning. Then, by treating the grids (*i.e.*, data patterns) as sampling units, BLAST employs grid sampling and grid mixup to sample the data in a balanced and comprehensive manner. As illustrated in Figure 3, BLAST involves several key processes: raw data construction, metrics calculation, feature construction, dimension reduction, and the sampling stage.

5.1 Raw Data Construction

We integrate extensive publicly available datasets, creating a large-scale dataset with a total of 321 billion observations. We fill missing values with zeros and filter out short time series (those with a length of less than 512). Commonly used benchmarks [47] are excluded. Furthermore, we apply z-score normalization to eliminate the influence of varying value ranges across datasets. See Appendix A.1 for more details.

5.2 Metrics Calculation

As a core component of BLAST, metrics calculation serves to characterize a time series through a diverse set of metrics. For a given

time series X, BLAST utilizes seven statistical metrics, which characterize a time series' patterns from various aspects [33]. Due to space limitations, additional details, including metrics selection principles, implementation details, and discussions on alternative methods, are provided in Appendix A.2.

Stationarity refers to whether the statistical properties of a time series remain constant over time. To assess this, we utilize the Augmented Dickey-Fuller (ADF) test, defined as:

$$Stationary = \begin{cases} True, & \text{if ADF}(X) < 0.05, \\ False, & \text{otherwise.} \end{cases} \quad (5)$$

The ADF test yields a boolean result, determining whether a given time series exhibits weak stationarity. Strong stationarity is not considered, as it is rarely encountered in real-world applications.

Trend describes the overall direction of change in a time series, reflecting long-term variation and representing a low-frequency component. To quantify the trend, we apply the Mann-Kendall test, formulated as:

$$Trend, Strength_t = \text{MannKendall}(X). \quad (6)$$

The *Trend* can be classified as either *increasing*, *decreasing*, or *no trend*, while $Strength_t$ is a floating-point value that quantifies the magnitude or significance of the detected trend.

Seasonality represents recurrent fluctuations within a time series, characterized by high-frequency components. We apply the Multiple Seasonal-Trend decomposition using Loess (M-STL) [4] to decompose the time series into residual (R), trend (T), and multiple seasonal components (S_i):

$$[S_1, \cdots, S_k], T, R = \text{M-STL}(X),$$
$$Strength_s = \max(0, 1 - \frac{\text{var}(R)}{\text{var}(R + \sum_1^k S_i)}), \quad (7)$$

where $Strength_s$ indicates the strength of seasonality. We use the number of seasonal components, denoted as k (with a maximum value of 3), along with $Strength_s$, as the metrics. Note that while the STL decomposition can also be used to calculate trends, doing so may result in redundancy between the trend and seasonality components, reducing their diversity.

Volatility quantifies the degree of fluctuation in a time series and is formally defined as:

$$Volatility = \frac{\sqrt{\frac{1}{T} \sum_{i=1}^{T} (x_i - \mu)^2}}{\mu}, \quad (8)$$

where μ is the mean of the time series with length T. Essentially, volatility is a variation of the standard deviation, reflecting the relative magnitude of variability.

Scedasticity indicates whether the variance of a time series changes over time, thereby capturing distribution drift. It can be assessed using Lagrange Multiplier (LM) test on the residual component [5]:

$$Scedasticity = \begin{cases} Homo, & \text{if LMTest}(R) > 0.05, \\ Hetero, & \text{otherwise.} \end{cases} \quad (9)$$

Memorability quantifies the degree of long-term dependence in a time series and is measured using the Hurst exponent:

$$Memorability = Hurst(X). \quad (10)$$

Anomaly represents the proportion of values that deviate significantly from the majority, reflecting the level of noise in the series. Outliers are identified as values exceeding the 95% threshold in a one-tailed test after z-score normalization:

$$Anomaly = \frac{|\{x_i \in X | \frac{x_i - \mu}{\sigma} > 1.645\}|}{T}. \quad (11)$$

5.3 Feature Construction

Overall, the metrics described above provide a comprehensive characterization of a time series. Figure 4 illustrates the distribution of the raw data across these metrics. As can be seen, these metrics are inherently heterogeneous, comprising both discrete and floating-point values with varying ranges. To mitigate this heterogeneity, we introduce a discretization-based feature construction approach that unifies the representation of all metrics into a single vector.

For continuous metrics, we discretize their values within a pre-defined range using a quantization technique. Formally, inspired by [3], given a metric z, the interval $[b_0, b_B]$ is divided into B equally spaced bins, and z is mapped to the corresponding bin index using the quantization function $g(z)$, defined as follows:

$$g(z) = \begin{cases} 0, & \text{if } b_0 \le z < b_1, \\ 1, & \text{if } b_1 \le z < b_2, \\ \vdots \\ B-1, & \text{if } b_{B-1} \le z < b_B. \end{cases} \quad (12)$$

Values outside the interval $[b_0, b_B]$ are assigned to the nearest bin (either 0 or $B-1$) to handle the long-tail distribution. The parameters B, b_0, and b_B for each continuous metric are listed in Table 4.

Finally, along with discrete metrics, we apply one-hot encoding to all metrics. These vectors are then concatenated into a unified representation h, which has a fixed length of 61, *i.e.*, a vector in $\mathbb{R}^{61}$, providing a standardized and comprehensive description of the time series patterns.

5.4 Dimension Reduction

To better understand the bias in the data distribution, we reduce the dimension of the vector h to a low-dimensional space. Specifically, for a given time series X_k^n, its corresponding vector h can be calculated. The BLAST raw dataset comprises approximately 40 million raw time series. Subsequently, we employ the UMAP [28] model f_{umap} to project all sparse vectors h into a dense two-dimensional space. Compared with other dimension reduction techniques such as t-SNE [39] and PCA [26], UMAP offers the advantages of higher efficiency and better preservation of data structure. The transformation is expressed as follows:

$$h' = f_{\text{umap}}(h) \in \mathbb{R}^2, \quad (13)$$

Table 4: Discretization of continuous metrics.

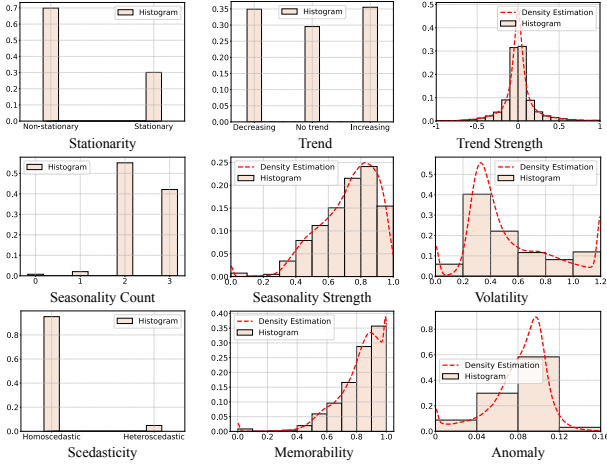| Metric	$Strength_t$	$Strength_s$	Volatility	Memorability	Anomaly
B	20	10	6	10	4
b_0	-1	0	0	0	0
b_B	1	1	1.2	1	0.16

Figure 4: Distribution of the raw dataset across key metrics.

where h represents the original vector, and h' is the corresponding vector after dimension reduction. We normalize all h' to [0, 1]. Due to space constraints, the details of the UMAP model's implementation and hyper-parameter study are provided in Appendix A.3.

As shown in Figure 3, the reduced data reveals a clear global structural pattern, though its distribution remains highly imbalanced. This skewed distribution can introduce bias during model training, as discussed in Section 1. Furthermore, the gaps between different regions suggest that the patterns in the raw dataset are still insufficient, despite the large scale of the data.

5.5 Sampling

To address the issue of uneven data distribution, we propose an intuitive and effective sampling approach, which incorporates both grid sampling and grid mixup.

First, we uniformly partition the two-dimensional space ($x, y \in$ [0, 1]) into $M \times M$ grids, denoted as $\mathcal{G}$, with each grid containing multiple time series. Grid sampling is then applied, which involves first selecting a grid, then randomly sampling a time series within that grid, followed by naive sampling. The probability of selecting a sample $W_{n,k,t}$ is given by the following:

$$\mathbb{P}(W_{n,k,t}) = \mathbb{P}(\mathcal{G}_m)\cdot\mathbb{P}(\mathcal{W}_{n,k} \mid \mathcal{G}_m) \cdot \mathbb{P}(W_{n,k,t} \mid \mathcal{W}_{n,k}),$$

$$\mathbb{P}(\mathcal{G}_m) = \frac{1}{|\mathcal{G}|},$$

$$\mathbb{P}(\mathcal{W}_{n,k} \mid \mathcal{G}_m) = \frac{1}{|\mathcal{G}_m|}, \mathbb{P}(W_{n,k,t} \mid \mathcal{W}_{n,k}) = \frac{1}{|\mathcal{W}_{n,k}|}, \forall X_k^n \in \mathcal{G}_m,$$

$$(14)$$

where $|\mathcal{G}|$ is the number of valid grids , $|\mathcal{G}_m|$ represents the number of time series included in m-th grid, and $\mathcal{W}_{n,k}$ is the candidate sample set generate from time series X_k^n. We set $M = 100$.

Next, to address the lack of sufficient coverage, *i.e.,* the gaps between different regions of the data distribution, we introduce a grid mixup technique that further enhances the model's generalization ability. Specifically, we randomly pick k grids (from all available grids), where k is drawn from the discrete uniform distribution $\mathcal{U}(1, K)$, and then randomly select samples from these grids. These

samples are subsequently mixed as follows:

$$X^{\text{GridMixup}} = \sum_{i=1}^{k} \lambda_i X^i, \quad (15)$$

where X^i is the sample from grid i, and $[\lambda_1, \cdots, \lambda_k]$ are sampled from a symmetric Dirichlet distribution $D(\alpha)$, where $\alpha = 1.5$. We set $K = 3$, *i.e.,* the original data remains in the dataset with a 33.33% probability. This approach is inspired by TSMixup [3], but instead of treating each time series as the basic unit of sampling, we use the grid as the fundamental sampling unit.

In summary, the sampling stage mitigates bias in over-dense or under-dense regions, effectively addressing biases in large-scale datasets. This strategy ensures that the samples are balanced and representative, thereby enhancing both the efficiency and generalization performance of the model training process.

6 Experiments

This section addresses the following key research questions through comprehensive experiments:

- **RQ1:** Does pre-training on BLAST provide any advantages?
- **RQ2:** What are the sources of these advantages, and what is the impact of different sampling strategies?
- **RQ3:** How do grid sampling and grid mixup influence balanced sampling (through ablation and hyperparameter analysis)?

6.1 Experimental Setup

6.1.1 **Baselines**. We select three popular universal forecasting models—TimeMoE [37], MOIRAI [43], and Chronos [3]. For TimeMoE and MOIRAI, we consider both their base and large versions, whereas for Chronos[3] we include the small and base versions. This yields six baselines in total. The dataset sizes and sampling methods originally used in their respective paper are detailed in Table 3.

6.1.2 **Datasets**. Following TimeMoE [37], we select six commonly used benchmarks: ETTh1, ETTh2, ETTm1, ETTm2, Weather, and GlobalTemp. None of these datasets is included in BLAST. Additionally, we adopt GIFT-Eval [2], the latest comprehensive benchmark containing 97 small prediction tasks. After filtering out any data present in Time-300B (TimeMoE pre-training data), LOTSA (MOIRAI pre-training data), and BLAST, we use the remaining 43 tasks based on the original GIFT-Eval settings.

6.1.3 **Implementation Details**. All experiments are conducted using PyTorch on 8×A100 GPUs (40GB). The code for training universal forecasting models with BLAST is available at *https://github.com/GestaltCogTeam/BasicTS*, and the BLAST corpus generation code can be found at *https://github.com/GestaltCogTeam/BLAST*. Additionally, all subsequent experimental results follow the *zero-shot* forecasting setting. Further implementation details for the benchmark datasets are provided in Appendix B.

6.2 Pre-training on BLAST (RQ1)

This section evaluates the advantages of training universal forecasting models using the BLAST corpus. To achieve this, we retrain each of the selected baselines, as detailed in §6.1.1, **from scratch**

---

[3]We employ the latest Chronos-Bolt release for its superior efficiency and accuracy.

Table 5: Performance comparison of BLAST retrained models with their pretrained counterparts. Lower MAE and MSE values indicate superior performance. The symbols s, b, and l represent the small, base, and large versions, respectively. † denotes the models retrained from scratch using the BLAST corpus. Models with superior or equal performance are highlighted in red.

Models	TimeMoE$_l^\dagger$		TimeMoE$_l$		TimeMoE$_b^\dagger$		TimeMoE$_b$		MOIRAI$_l^\dagger$		MOIRAI$_l$		MOIRAI$_b^\dagger$		MOIRAI$_b$		Chronos$_b^\dagger$		Chronos$_b$		Chronos$_s^\dagger$		Chronos$_s$	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1 96	.348	.375	.350	.382	.352	.376	.357	.381	.359	.383	.381	.388	.362	.384	.376	.392	.357	.375	.384	.379	.359	.376	.394	.381
192	.381	.399	.388	.412	.389	.401	.384	.404	.395	.404	.434	.415	.400	.405	.412	.413	.397	.401	.441	.412	.403	.402	.455	.414
336	.409	.424	.411	.430	.408	.419	.411	.434	.411	.416	.495	.445	.416	.420	.433	.428	.422	.417	.475	.430	.431	.416	.499	.444
720	.447	.451	.427	.455	.450	.455	.449	.477	.420	.430	.611	.510	.430	.439	.447	.444	.460	.443	.472	.446	.449	.439	.520	.476
AVG	.396	.412	.394	.419	.399	.412	.400	.424	.396	.408	.480	.439	.402	.412	.417	.419	.409	.409	.443	.416	.410	.408	.467	.428
ETTh2 96	.276	.329	.302	.354	.285	.332	.305	.359	.288	.325	.296	.330	.284	.324	.294	.325	.282	.321	.289	.330	.281	.326	.282	.328
192	.345	.376	.364	.385	.348	.378	.351	.386	.353	.370	.361	.371	.348	.369	.365	.375	.356	.369	.359	.369	.353	.371	.354	.373
336	.384	.416	.417	.425	.372	.405	.391	.418	.369	.382	.390	.390	.367	.386	.376	.390	.378	.397	.399	.400	.387	.403	.416	.410
720	.442	.470	.537	.496	.419	.452	.419	.454	.387	.406	.423	.418	.387	.410	.416	.433	.403	.424	.420	.425	.411	.430	.428	.431
AVG	.361	.397	.405	.415	.356	.391	.366	.404	.349	.370	.367	.377	.346	.372	.362	.382	.355	.377	.366	.381	.358	.382	.370	.385
ETTm1 96	.327	.343	.309	.357	.334	.350	.338	.368	.355	.355	.380	.361	.348	.354	.363	.356	.310	.327	.331	.333	.314	.331	.328	.332
192	.368	.378	.346	.381	.388	.386	.353	.388	.388	.380	.412	.383	.385	.378	.388	.375	.363	.360	.386	.365	.364	.365	.365	.384
336	.373	.396	.373	.408	.400	.412	.381	.413	.399	.387	.436	.400	.410	.394	.416	.392	.410	.387	.408	.382	.391	.417	.391	.425
720	.445	.438	.475	.477	.457	.451	.504	.493	.429	.413	.462	.420	.448	.416	.460	.418	.477	.427	.503	.430	.452	.521	.445	.525
AVG	.378	.388	.375	.405	.394	.399	.394	.415	.392	.383	.422	.391	.397	.385	.406	.385	.390	.375	.407	.377	.380	.408	.382	.416
ETTm2 96	.180	.259	.197	.286	.181	.260	.201	.291	.192	.259	.211	.274	.194	.265	.205	.273	.175	.249	.177	.244	.180	.248	.180	.251
192	.245	.305	.250	.322	.247	.307	.258	.334	.256	.302	.281	.318	.257	.304	.275	.316	.242	.290	.251	.293	.243	.292	.251	.298
336	.283	.338	.337	.375	.293	.344	.324	.373	.289	.329	.341	.355	.301	.342	.329	.350	.299	.326	.305	.327	.302	.331	.315	.338
720	.364	.392	.480	.461	.376	.396	.488	.464	.372	.384	.428	.428	.387	.396	.437	.411	.394	.387	.419	.394	.406	.396	.421	.403
AVG	.268	.323	.316	.361	.274	.326	.317	.365	.277	.318	.315	.343	.284	.326	.311	.337	.277	.313	.288	.314	.282	.316	.291	.330
Weather 96	.161	.209	.159	.213	.163	.213	.160	.214	.168	.200	.278	.376	.171	.202	.220	.217	.163	.197	.177	.210	.164	.198	.172	.206
192	.217	.261	.215	.266	.215	.263	.210	.260	.246	.217	.301	.409	.218	.247	.271	.259	.210	.241	.224	.253	.213	.244	.218	.248
336	.276	.304	.291	.322	.273	.297	.309	.309	.288	.275	.329	.420	.278	.291	.286	.297	.264	.282	.260	.276	.273	.288	.266	.282
720	.342	.353	.415	.400	.328	.339	.418	.405	.351	.375	.370	.463	.370	.350	.373	.354	.339	.334	.345	.331	.349	.342	.358	.339
AVG	.249	.281	.270	.300	.244	.278	.274	.297	.263	.266	.319	.417	.259	.272	.287	.281	.244	.263	.251	.267	.249	.268	.253	.268
GlobalTemp 96	.226	.346	.219	.341	.229	.349	.230	.350	.234	.351	.278	.376	.236	.352	.273	.377	.226	.343	.236	.352	.230	.345	.233	.348
192	.263	.386	.265	.381	.272	.390	.268	.385	.266	.382	.301	.409	.268	.384	.304	.409	.271	.384	.287	.398	.280	.389	.287	.397
336	.309	.420	.326	.426	.311	.423	.326	.427	.309	.420	.329	.420	.309	.420	.332	.437	.314	.419	.332	.433	.318	.420	.320	.430
720	.340	.447	.344	.453	.343	.449	.377	.467	.361	.459	.379	.467	.347	.449	.379	.469	.427	.502	.463	.524	.438	.504	.452	.521
AVG	.284	.399	.288	.400	.288	.402	.300	.407	.292	.403	.321	.418	.290	.401	.322	.423	.322	.418	.329	.426	.316	.414	.323	.424
# Wins	22	28	9	2	23	28	9	2	30	30	0	0	30	28	0	3	28	26	2	5	28	28	4	3

Table 6: Performance comparison on the GIFT-Eval benchmark. Data previously included in Time-300B, LOTSA, and BLAST have been excluded. Lower MASE values indicate better performance. Models with superior performance are highlighted in red.

Models	TimeMoE$_l^\dagger$	TimeMoE$_l$	TimeMoE$_b^\dagger$	TimeMoE$_b$	MOIRAI$_l^\dagger$	MOIRAI$_l$	MOIRAI$_b^\dagger$	MOIRAI$_b$	Chronos$_b^\dagger$	Chronos$_b$	Chronos$_s^\dagger$	Chronos$_s$
MASE	0.777	0.872	0.760	0.888	0.740	0.816	0.759	0.812	0.711	0.740	0.738	0.742

using the BLAST corpus. We then compare the performance of the retrained models with their pre-trained counterparts on the benchmarks outlined in §6.1.2.

Settings. We adhered to the original setup as outlined in their respective papers [3, 37, 43]. Due to space limitations, readers interested in more details can refer to the original papers. The only deviation from the original setup was the batch size for TimeMoE. The original TimeMoE model [37] was trained using **128×A100 GPUs** with a **batch size of 1024**, processing **419.43 billion training tokens**. Thanks to its massive model parameters and training

data, it achieved state-of-the-art performance. However, due to computational resource constraints, the TimeMoE model pre-trained on BLAST used a reduced **batch size of 192**, training on **78.64 billion tokens**. For the benchmarks used in TimeMoE [37], we follow a similar setup. We assess the performance across four different prediction lengths: [96, 192, 336, 720]. We report the normalized Mean Squared Error (MSE) and Mean Absolute Error (MAE). For the GIFT-Eval benchmark [2], we filtered out data already included in Time-300B (TimeMoE pre-training data), LOTSA (MOIRAI pre-training data), and BLAST, and strictly followed its evaluation pipeline. We report the Mean Absolute Scaled Error (MASE).

Table 7: Performance of different sampling strategies. Models with superior performance are highlighted in red.

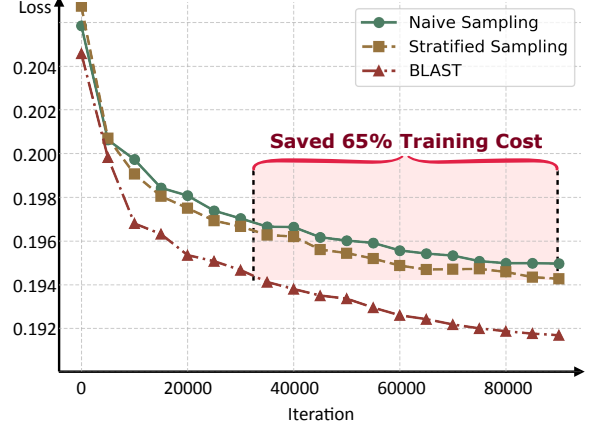|	Naive Sampling				Stratified Sampling				Balanced Sampling				w/o GS				w/o GM			
Horizons	96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720
ETTh1	0.393	0.421	0.468	0.507	0.388	0.412	0.458	0.503	0.376	0.401	0.419	0.455	0.390	0.418	0.448	0.489	0.386	0.414	0.444	0.483
ETTh2	0.364	0.401	0.460	0.513	0.362	0.399	0.458	0.495	0.332	0.378	0.405	0.452	0.366	0.401	0.455	0.500	0.338	0.388	0.422	0.465
ETTm1	0.379	0.422	0.454	0.501	0.372	0.413	0.450	0.494	0.350	0.386	0.412	0.451	0.366	0.412	0.450	0.485	0.355	0.389	0.435	0.474
ETTm2	0.303	0.344	0.381	0.419	0.299	0.330	0.376	0.409	0.260	0.307	0.344	0.396	0.305	0.338	0.370	0.403	0.275	0.318	0.353	0.400

Results. Table 5 and Table 6 present the results of our experiments. In general, models pre-trained on the BLAST corpus outperform the original models. The results for TimeMoE highlight the significant efficiency advantages brought by pre-training on BLAST, both in terms of computational resources and data usage. Specifically, BLAST-based pre-training requires only 8 A100 GPUs, compared to 128 A100 GPUs for the original TimeMoE, and processes 78.64 billion training tokens, which is a fraction of the 419.43 billion tokens required for the original model. Furthermore, the results for MOIRAI and Chronos demonstrate that, when computational resources and the number of training tokens are similar, the performance advantages brought by BLAST become even more apparent. In the next part, we delve deeper into the impact of sampling strategies on both training efficiency and model performance.

6.3 Impact of Sampling Strategies (RQ2)

This section provides a comprehensive analysis of the impact of different sampling strategies on **training efficiency** and **predictive performance**, shedding light on the key factors contributing to the advantages of BLAST pre-training. To quantify the effects of each sampling strategy precisely, we conduct controlled experiments **using the same raw data.**

Settings. We use TimeMoE$_{base}$ as the baseline model, with experimental configurations consistent with §6.2. Based on the raw BLAST data, TimeMoE$_{base}$ are trained using datasets derived from three different sampling strategies: naive sampling (§4.1), stratified sampling (§4.2), and balanced sampling (§5). First, to evaluate the effects of these strategies on training efficiency, we analyze the rate of validation loss reduction on a unified validation set, which is constructed as the union of the validation sets from the three sampling strategies and excludes data that appears in the training sets. Second, to assess the impact of sampling strategies on model performance, we report the MAE on four ETT datasets, enabling a comprehensive comparison across different sampling methods.

Results. Figure 5 illustrates the convergence rates of models trained with different sampling strategies, highlighting their effects on training efficiency. The results indicate that models trained with balanced sampling exhibit a significantly faster reduction in loss. This efficiency advantage becomes particularly evident in the later stages of training, where loss reduction slows. Notably, under equivalent loss conditions, balanced sampling requires only about 35% of the training steps compared to naive or stratified sampling. Table 7 further demonstrates the effectiveness of different sampling methods in terms of forecasting performance. Balanced sampling consistently outperforms other methods. Additionally,

Figure 5: Comparison of convergence speeds for different sampling methods.

models trained with naive or stratified sampling underperform the original TimeMoE. This is due to the lack of focus on data diversity in these strategies, combined with the substantially smaller token count in our training process compared to the original TimeMoE implementation.

In summary, these results underscore the critical role of data diversity, and the balanced sampling strategy significantly enhances data diversity during training. This intuitive yet effective sampling approach proves instrumental in improving both model performance and training efficiency.

6.4 How Do Grid Sampling and Grid Mixup Affect Balanced Sampling? (RQ3)

This part presents a further ablation study and hyper-parameter analysis of BLAST. Specifically, we examine the contributions of two key components—grid sampling and grid mixup—in balanced sampling. Additionally, we investigate how grid size affects model performance and explore the underlying reasons for these effects.

Settings. We use TimeMoE$_{base}$ as the baseline model and conduct experiments on datasets excluding either grid sampling or grid mixup. We report the MAE on four ETT datasets. Furthermore, we vary the grid size in the sampling stage, setting it to $[10, 50, 100, 500, 1000, 5000]$. We evaluate the models on four ETT datasets and report their averaged predictive performance.

Results. The performance of models without grid sampling and grid mixup is shown in Table 7. Removing grid sampling results in a setup similar to naive sampling, where grid mixup becomes a
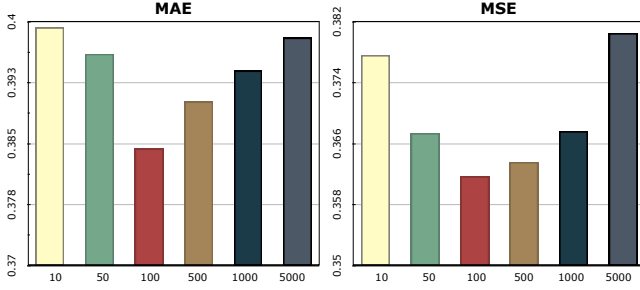
Figure 6: The impact of grid size in grid sampling.

Table 8: Comparison between PCA, t-SNE, and UMAP.

standard TSMixup [3]. This yields slightly better performance than naive sampling. Meanwhile, the absence of grid mixup significantly diminishes performance compared to balanced sampling. This confirms that grid mixup is an effective strategy for enhancing data diversity. These ablation results further validate the effectiveness of balanced sampling.

Additionally, Figure 6 presents the predictive performance for various grid sizes. It is evident that both excessively small and large grids result in suboptimal performance. The reaseaons are:

- Too large a grid: Results in too few grids, each with many heterogeneous time series. In the extreme case, there's just one grid, and balanced sampling degrades to naive sequence sampling.
- Too small a grid: Results in too many grids. Despite large data, the representation space remains sparse, and balanced sampling degrades to naive sequence sampling again due to insufficient sequences per grid.

In summary, grid size acts like implicit clustering—ineffective clustering (either too large or too small grid size) causes balanced sampling to fail.

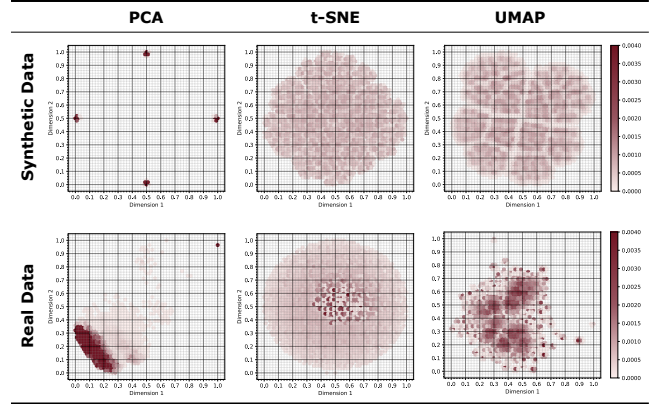## 6.5 Alternative Dimension Reduction Methods

We benchmark three popular dimensionality-reduction algorithms, PCA [26], t-SNE [39], and UMAP. To obtain an intuitive sanity check, we generated synthetic data with uniformly distributed feature vectors, following the feature construction process in BLAST. If a method faithfully preserves the original geometry, its projection should therefore exhibit:

- Clear global structure, as the unified vector is constructed from multiple one-hot vectors.
- Even distribution of samples within each component, as each one-how vector is randomly generated.

To compare these methods, we visualized the results for both real and synthetic data.

Table 8 contrasts the three dimensionality-reduction methods. Because PCA is a linear method, it fails to represent either local or global structure in our discrete feature vectors. t-SNE preserves neighbourhoods but distorts the overall geometry and, on large datasets, is computationally heavy. UMAP, by comparison, captures both global relationships and local patterns: it separates the main regions cleanly and keeps the samples within each region evenly distributed. Overall, UMAP provides the most faithful picture of the data at both macro- and micro-scales.

6.6 Intuition Behind Balanced Sampling

Essentially, BLAST estimates the probability density function (PDF) of the pre-training data and then draws unbiased samples via stratified sampling guided by that PDF. Directly estimating a PDF for raw time series data is impractical because each series is a high-dimensional vector. BLAST therefore compresses every series into a small set of statistical descriptors and projects these descriptors into a two-dimensional feature space. It then partitions this plane into uniform grid cells and samples within them. Each cell implicitly defines a cluster—capturing a characteristic pattern—so the cells themselves, rather than pre-defined classes or domain labels, become the strata for sampling.

Additionally, explicit clustering algorithms such as k-means or DBSCAN could serve the same purpose, but they scale poorly and often fail to preserve cluster quality on large datasets. Grid sampling offers a more intuitive, computationally lightweight alternative that strikes a practical balance between simplicity and effectiveness.
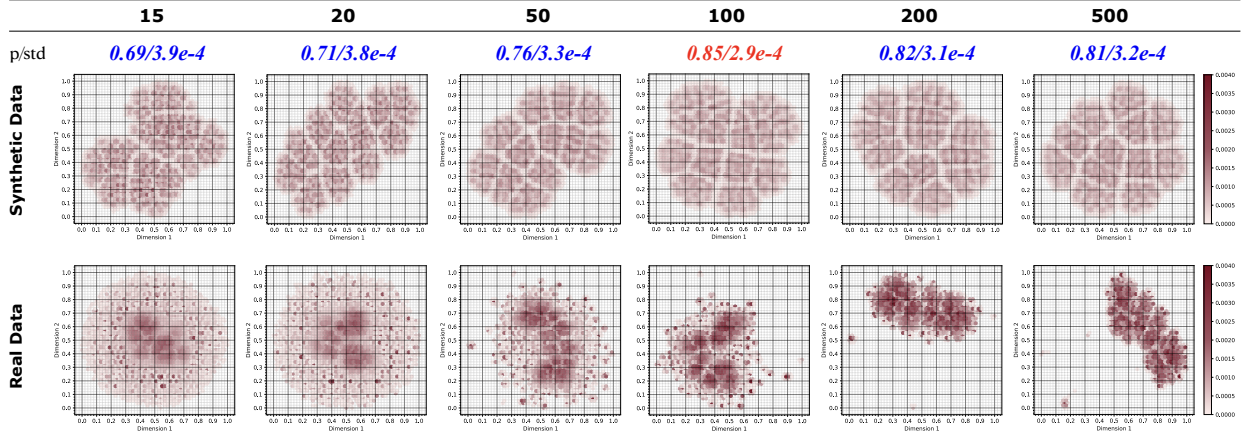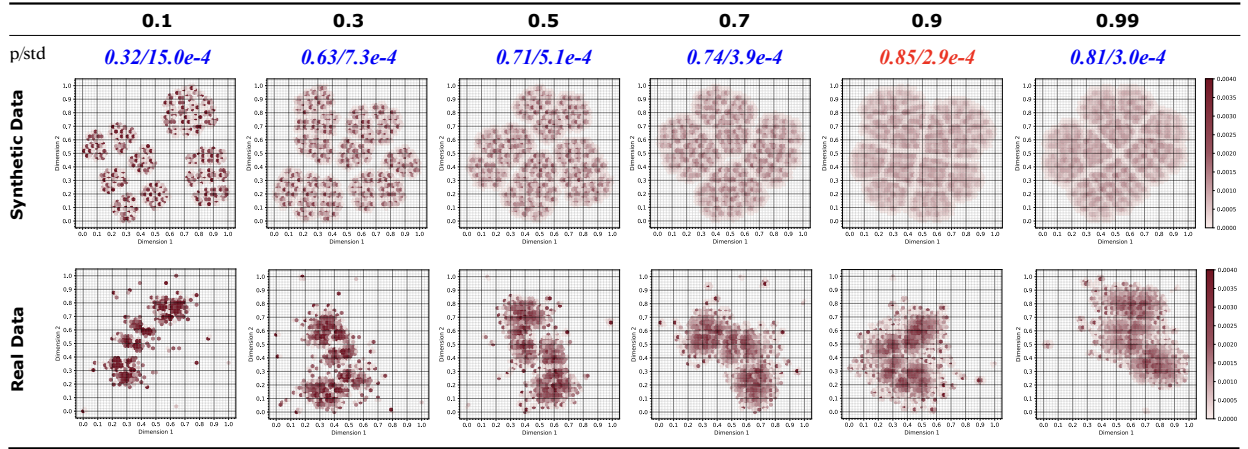
7 Conclusion

In this work, we present BLAST, a balanced sampling time series corpus designed to address the critical yet understudied challenge of data diversity in training universal forecasting models. By integrating 321 billion observations from diverse public datasets and introducing a novel balanced sampling strategy, BLAST systematically mitigates inherent biases in large-scale time series distributions. The proposed balanced sampling techniques ensure representative pattern coverage, thereby enhancing both the training efficiency and generalization capability of the model. Extensive experiments demonstrate that models pre-trained on BLAST achieve superior zero-shot forecasting accuracy, outperforming models trained on naively or stratified-sampled corpora.

Acknowledgments

References

[1] Amro Kamal Mohamed Abbas, Kushal Tirumala, Daniel Simig, Surya Ganguli, and Ari S Morcos. [n. d.]. SemDeDup: Data-efficient learning at web-scale through semantic deduplication. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

[2] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. 2024. GIFT-Eval: A Benchmark For General Time Series Forecasting Model Evaluation. *arXiv preprint arXiv:2410.10393* (2024).

[3] 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. 2024. Chronos: Learning the Language of Time Series. *Transactions on Machine Learning Research* (2024).

[4] Kasun Bandara, Rob J Hyndman, and Christoph Bergmeir. 2021. MSTL: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns. *arXiv preprint arXiv:2107.13462* (2021).

[5] Tim Bollerslev. 1986. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics* 31, 3 (1986), 307–327.

[6] Mouxiang Chen, Lefei Shen, Zhuo Li, Xiaoyun Joy Wang, Jianling Sun, and Chenghao Liu. 2024. Visionts: Visual masked autoencoders are free-lunch zero-shot time series forecasters. *arXiv preprint arXiv:2408.17253* (2024).

[7] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. 2015. The UCR Time Series Classification Archive. www.cs.ucr.edu/~eamonn/time_series_data/.

[8] Luke Nicholas Darlow, Qiwen Deng, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Artjom Joosen, Adam Barker, and Amos Storkey. 2024. DAM: Towards a Foundation Model for Forecasting. In *The Twelfth International Conference on Learning Representations*.

[9] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. [n. d.]. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*.

[10] Jinliang Deng, Xiusi Chen, Renhe Jiang, Du Yin, Yi Yang, Xuan Song, and Ivor W Tsang. 2024. Disentangling structured components: Towards adaptive, interpretable and scalable time series forecasting. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[11] Jinliang Deng, Feiyang Ye, Du Yin, Xuan Song, Ivor Tsang, and Hui Xiong. 2024. Parsimony or capability? decomposition delivers both in long-term time series forecasting. *Advances in Neural Information Processing Systems* 37 (2024), 66687–66712.

[12] Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha Naidu, and Colin White. 2023. ForecastPFN: synthetically-trained zero-shot forecasting. In *NeurIPS*.

[13] Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. 2024. Tiny Time Mixers (TTMs): Fast Pre-trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series. In *NeurIPS*.

[14] Shanghua Gao, Teddy Koker, Owen Queen, Thomas Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. 2024. UniTS: A Unified Multi-Task Time Series Model. In *NeurIPS*.

[15] Azul Garza and Max Mergenthaler-Canseco. 2023. TimeGPT-1. *arXiv preprint arXiv:2310.03589* (2023).

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

[17] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. [n. d.]. MOMENT: A Family of Open Time-series Foundation Models. In *Forty-first International Conference on Machine Learning*.

[18] Jincai Huang, Yongjun Xu, Qi Wang, Qi Cheems Wang, Xingxing Liang, Fei Wang, Zhao Zhang, Wei Wei, Boxuan Zhang, Libo Huang, et al. 2025. Foundation models and intelligent decision-making: Progress, challenges, and perspectives. *The Innovation* (2025).

[19] Chenxi Liu, Hao Miao, Qianxiong Xu, Shaowen Zhou, Cheng Long, Yan Zhao, Ziyue Li, and Rui Zhao. 2025. Efficient Multivariate Time Series Forecasting via Calibrated Language Models with Privileged Knowledge Distillation. In *ICDE*.

[20] Chenxi Liu, Qianxiong Xu, Hao Miao, Sun Yang, Lingzheng Zhang, Cheng Long, Ziyue Li, and Rui Zhao. 2025. Timecma: Towards llm-empowered multivariate time series forecasting via cross-modality alignment. In *AAAI*, Vol. 39.

[21] Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. 2024. Moirai-MoE: Empowering Time Series Foundation Models with Sparse Mixture of Experts. *arXiv preprint arXiv:2410.10469* (2024).

[22] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024. 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.

[23] Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. 2024. Timer-XL: Long-Context Transformers for Unified Time Series Forecasting.

[24] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. [n. d.]. Timer: Generative Pre-trained Transformers Are Large Time Series Models. In *Forty-first International Conference on Machine Learning*.

[25] Zhiding Liu, Jiqian Yang, Mingyue Cheng, Yucong Luo, and Zhi Li. 2024. Generative pretrained hierarchical transformer for time series forecasting. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

[26] Andrzej Maćkiewicz and Waldemar Ratajczak. 1993. Principal components analysis (PCA). *Computers & Geosciences* 19, 3 (1993), 303–342.

[27] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2022. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting* 38, 4 (2022), 1346–1364.

[28] Leland McInnes and John Healy. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *CoRR* abs/1802.03426 (2018). arXiv:1802.03426

[29] Hao Miao, Ziqiao Liu, Yan Zhao, Chenjuan Guo, Bin Yang, Kai Zheng, and Christian S Jensen. 2024. Less is more: Efficient time series dataset condensation via two-fold modal matching. *PVLDB* 18, 2 (2024), 226–238.

[30] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *ICLR*. OpenReview.net.

[31] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023). arXiv:2303.08774

[32] John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S Tsay, Themis Palpanas, and Michael J Franklin. 2022. Tsb-uad: an end-to-end benchmark suite for univariate time-series anomaly detection. *Proceedings of the VLDB Endowment* 15, 8 (2022), 1697–1711.

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

[34] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Hassen, Anderson Schneider, et al. 2023. Lag-llama: Towards foundation models for time series forecasting. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

[35] Yunfan Shao, Linyang Li, Zhaoye Fei, Hang Yan, Dahua Lin, and Xipeng Qiu. 2024. Balanced Data Sampling for Language Model Training with Clustering. *arXiv preprint arXiv:2402.14526* (2024).

[36] Zezhi Shao, Tangwen Qian, Tao Sun, Fei Wang, and Yongjun Xu. 2025. Spatial-temporal large models: A super hub linking multiple scientific areas with artificial intelligence. *The Innovation* 6, 2 (2025).

[37] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. 2024. Time-MoE: Billion-Scale Time Series Foundation Models with Mixture of Experts. *arXiv preprint arXiv:2409.16040* (2024).

[38] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023).

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

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*.

[41] Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. 2025. ChatTime: A Unified Multimodal Time Series Foundation Model Bridging Numerical and Textual Data. In *AAAI Conference on Artificial Intelligence*.

[42] Yihang Wang, Yuying Qiu, Peng Chen, Kai Zhao, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. 2024. ROSE: Register Assisted General Time Series Forecasting with Decomposed Frequency Learning. *arXiv preprint arXiv:2405.17478* (2024).

[43] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. [n. d.]. Unified Training of Universal Time Series Forecasting Transformers. In *Forty-first International Conference on Machine Learning*.

[44] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *ICLR*. OpenReview.net.

[45] Haixu Wu, Hang Zhou, Mingsheng Long, and Jianmin Wang. 2023. Interpretable weather forecasting for worldwide stations with a unified deep model. *Nature Machine Intelligence* 5, 6 (2023), 602–611.

[46] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting? *AAAI*.

[47] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI*. 11106–11115.

Table 9: Hyperparameter study for n_neighbor.

	15	20	50	100	200	500
p/std	*0.69/3.9e-4*	*0.71/3.8e-4*	*0.76/3.3e-4*	*0.85/2.9e-4*	*0.82/3.1e-4*	*0.81/3.2e-4*

Table 10: Hyperparameter study for min_dist.

	0.1	0.3	0.5	0.7	0.9	0.99
p/std	*0.32/15.0e-4*	*0.63/7.3e-4*	*0.71/5.1e-4*	*0.74/3.9e-4*	*0.85/2.9e-4*	*0.81/3.0e-4*

A Details of BLAST

A.1 Raw Data Construction

Building upon previous work [3, 7, 17, 32, 43], we have collected a large-scale time series dataset reaching 321 billion data points. It is important to note that not all data were used for training. Common benchmark datasets, such as ETT, Weather, and Traffic, were excluded to ensure the integrity of our experimental settings. Additionally, we filtered out time series with more than 5% missing values (NaN). Moreover, we retained the remaining NaN values within the filtered time series. These missing values are handled dynamically during the training phase, according to the specific requirements of the model.

A.2 Metrics Calculation

A.2.1 **Selection Principles for Metrics**. Metrics selection is essential for effectively capturing the underlying patterns of a time series. The seven metrics selected in this study are widely used in statistical time series analysis, and each highlight different dynamic aspects, providing a comprehensive and complementing representation of the series' pattern. For example, trends and seasonality capture distinct components: trends represent low-frequency, long-term variations, while seasonality reflects high-frequency, periodic fluctuations. Stability, volatility, hetero/homo-scedasticity, and anomalies present distributional characteristics and variability from different angles. Furthermore, the combination of memory and seasonality could reveal the long-term dependency structure within the data. Additionally, it is crucial that *these metrics should not be directly tied to predictability*; otherwise, harmful samples may be introduced during the grid sampling process.

A.2.2 **Handling Variable-Length Series**. Although these metrics do not have stringent requirements on time series length, excessively long samples may result in less robust representations. Therefore, we standardize time series to a maximum context length of 4096. Specifically, for time series longer than 4096, we randomly sample three segments and compute the metrics for each. For continuous metrics, we take the average, while for discrete metrics, we use a voting strategy to select the most frequent value.

A.2.3 **Alternative Methods Considered**. The core objective of metrics calculation is to comprehensively capture the patterns of a time series. Any method capable of achieving this goal can be applied at this stage. One potential alternative is using deep learning models to generate time series representations. However, the raw

Table 11: Performance comparison of TimeMoE pre-trained on BLAST against full-shot models. Red: the best, Blue: 2nd best.

Models	Pre-training on BLAST				Full-shot Models											
	TimeMoE$_{base}$		TimeMoE$_{large}$		iTransformer		TimeMixer		TimesNet		PatchTST		TiDE		DLinear	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.399	0.412	0.396	0.412	0.454	0.447	0.448	0.442	0.454	0.450	0.468	0.454	0.540	0.507	0.455	0.451
ETTh2	0.356	0.391	0.361	0.397	0.383	0.406	0.364	0.395	0.414	0.496	0.386	0.406	0.611	0.549	0.558	0.515
ETTm1	0.394	0.399	0.378	0.388	0.407	0.409	0.381	0.395	0.400	0.405	0.387	0.400	0.419	0.419	0.403	0.406
ETTm2	0.274	0.326	0.268	0.323	0.288	0.332	0.275	0.323	0.291	0.332	0.280	0.326	0.358	0.403	0.350	0.400
Weather	0.244	0.278	0.249	0.281	0.257	0.278	0.240	0.271	0.258	0.286	0.258	0.280	0.270	0.320	0.265	0.316
Average	0.333	0.361	0.330	0.36	0.357	0.374	0.341	0.365	0.363	0.393	0.355	0.373	0.439	0.439	0.406	0.417

BLAST dataset is vast, containing 40 million time series, and there is currently no widely recognized and robust model for time series representation that can process such large-scale data efficiently. Additionally, while using statistical metrics to characterize a time series is significantly faster than deep learning models, it still requires considerable time. In our experiments, this process took 8 days using 128 CPU threads (Intel Xeon 6338 2.0GHz). Therefore, improving the efficiency of time series representation in the balanced sampling process remains a critical topic for future research.

A.3 UMAP Hyperparameter Study

A.3.1 UMAP Hyperparameter Description. The choice of UMAP parameters significantly impacts dimension reduction. In this study, the primary goal is to preserve the global structure of the large-scale dataset, particularly its overall distribution. Key UMAP parameters include n_neighbors, min_dist, and metric, which influence different aspects of the embedding process.

- **n_neighbors**: This parameter controls the balance between local and global structures. Larger values better capture the global distribution by considering more neighbors.
- **min_dist**: This determines the compactness of points in the reduced space. A higher value prevents excessive clustering of points, prioritizing the preservation of global topology.
- **metric**: This defines the distance function for measuring point similarity. Given the discretization process in BLAST, we use the Hamming distance, calculated as $d_H(x, y) = \sum_i \mathbb{I}(x_i \neq y_i)$, where x and y are two feature vectors, and $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the condition is true and 0 otherwise.

A.3.2 Hyperparameter Optimization. Similar to § 6.5, to optimize UMAP parameters, we generated synthetic data with uniformly distributed feature vectors, following the feature construction process in BLAST. Specifically, each one-hot feature was uniformly assigned across categories.

We assessed parameter effectiveness using two metrics: the **proportion of non-empty grids** (p) and the **standard deviation of grid density** (std) after dimension reduction. Larger p and smaller std indicate better results. Using n_neighbors = 100 and min_dist = 0.9 as the baseline, we tested values for n_neighbors in the range [15, 20, 50, 100, 200, 500] and for min_dist in [0.1, 0.3, 0.5, 0.7, 0.9, 0.99]. The results, shown in Table 9 and Table 10, show that the

dimension reduction is optimal when n_neighbors = 100 and min_dist = 0.9, with consistent trends observed for both real and synthetic datasets.

A.4 Using the BLAST Corpus

To facilitate user access, we directly provide the processed data. These datasets are represented as $N \times L$ matrices, where N denotes the number of samples, and L is the length of each sample. The length L is set to 4096, and samples shorter than 4096 are right-padded with NaN values to ensure uniform length. This approach allows users to easily read and utilize the samples.

B Details of Experiments

B.1 Evaluation Metrics

In this study, we use the Mean Absolute Error (MAE) and Mean Squared Error (MSE) as evaluation metrics. These metrics are commonly used to assess the performance of predictive models and can be formulated as follows:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|, \qquad \text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \qquad (16)$$

where y_i is the true value, $\hat{y}_i$ is the predicted value, and N is the total number of samples.

B.2 Details for Benchmark Datasets.

The ETTh1, ETTh2, ETTm1, ETTm2, and Weather datasets adhere to the standard settings established in previous studies. For evaluation, we utilize the test set for zero-shot prediction. The results we obtained are consistent with those reported in TimeMoE [37]. For the GlobalWeather dataset, since TimeMoE [37] does not follow conventional settings and lacks detailed descriptions, we perform the evaluation using the test set [45], applying z-score normalization and setting the stride S equal to the prediction length. For the GIFT-Eval benchmark, we follow its original setting.

B.3 Additional Results

We compare TimeMoE pretrained on BLAST with full-shot models [10, 11, 22, 30, 44, 46] on the ETTh1, ETTh2, ETTm1, ETTm2, and Weather datasets. Following the experimental settings in TimeMoE [37], we report the average error in Table 11. It can be observed that BLAST-pretrained TimeMoE outperforms these full-shot models.