MF-CLR: Multi-Frequency Contrastive Learning Representation for Time Series

Jufang Duan [1] **Wei Zheng** [1] **Yangzhou Du** [1] **Wenfa Wu** [1] **Haipeng Jiang** [1] **Hongsheng Qi** [1]

Abstract

Learning a decent representation from unlabeled time series is a challenging task, especially when the time series data is derived from diverse channels at different sampling rates. Our motivation stems from the financial domain, where sparsely labeled covariates are commonly collected at different frequencies, *e.g.*, daily stock market index, monthly unemployment rate and quarterly net revenue of a certain listed corporation. This paper presents **M**ulti-**F**requency **C**ontrastive **L**earning **R**epresentation (MF-CLR), aimed at learning a good representation of multi-frequency time series in a self-supervised paradigm by leveraging the ability of contrastive learning. MF-CLR introduces a hierarchical mechanism that spans across different frequencies along the feature dimension. Within each contrastive block, two groups of subseries with adjacent frequencies are embedded based on our proposed cross-frequency consistency. To validate the effectiveness of MF-CLR, we conduct extensive experiments on five downstream tasks, including long-term and short-term forecasting, classification, anomaly detection and imputation. Experimental evidence shows that MF-CLR delivers a leading performance in all the downstream tasks and keeps consistent performance across different target dataset scales in the transfer learning scenario. The source code is publicly available at *https://github.com/duanjufang/MF-CLR*.

1. Introduction

Time series data plays an increasingly significant role in various industries, *e.g.*, finance (Houssein et al., 2022; Zhao et al., 2023), supply chain (Punia et al., 2020; Punia & Shankar, 2022) and health care (Qin et al., 2023; Saveliev & van der Schaar, 2023). How to extract usable information from time series data and use them for the downstream task is a long run interest in the artificial intelligence community. As what is illustrated by previous works, the performance of AI methods depends heavily on data representations (Bengio et al., 2013). Inspired by the success in domains like computer vision (CV) (Chen et al., 2020) and natural language processing (NLP) (Gao et al., 2021), contrastive representation learning becomes more and more well-studied in time series realm. Approaches evolve from the naïve inspirations by experiences in CV or NLP domains (Franceschi et al., 2019; Eldele et al., 2021) to the ones that are more capable of depicting temporal dependencies (Woo et al., 2022; Zhang et al., 2022). However, unlike NLP domains (which have discrete signal space to build tokenized dictionaries) and CV domains (which have strong inductive bias such as cropping invariance) where researchers pursue a universal solution, representation learning in time series domains are more problem-oriented with enlightenment of domain knowledge (Mohsenvand et al., 2020; Tonekaboni et al., 2021; Sarkar & Etemad, 2022) or expert features (Nonnenmacher et al., 2022).

The motivation of our work is derived from real-world problems in the financial realm, where the ability to learn a decent representation from sparsely labeled data is crucial. Aside from the historical observations and covariates, many publicly available data can be collected to compensate for the shortage of input features. These include macroeconomic indicators, financial reports, and stock market trading information. Normally, the aforementioned data covers a much longer horizon [1] than the historical observations do. Under this circumstance, a self-supervised representation learning model can be trained on the unlabeled part of these fetched external data, and then the learned knowledge representation can be applied to the specific downstream task. However, different sources of external data may be sampled at different frequencies (we use the term *frequency* to refer to the sampling rate in the rest of this paper), *e.g.*, the GDP that is revealed monthly while the net income of a listed company that is revealed quarterly. Addressing

[1]Lenovo Research, Beijing, China. Correspondence to: Jufang Duan <duanjf2@lenovo.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]For example, 13 datasets of fiscal data dating back before 2000 can be found in https://fiscaldata.treasury.gov/datasets/

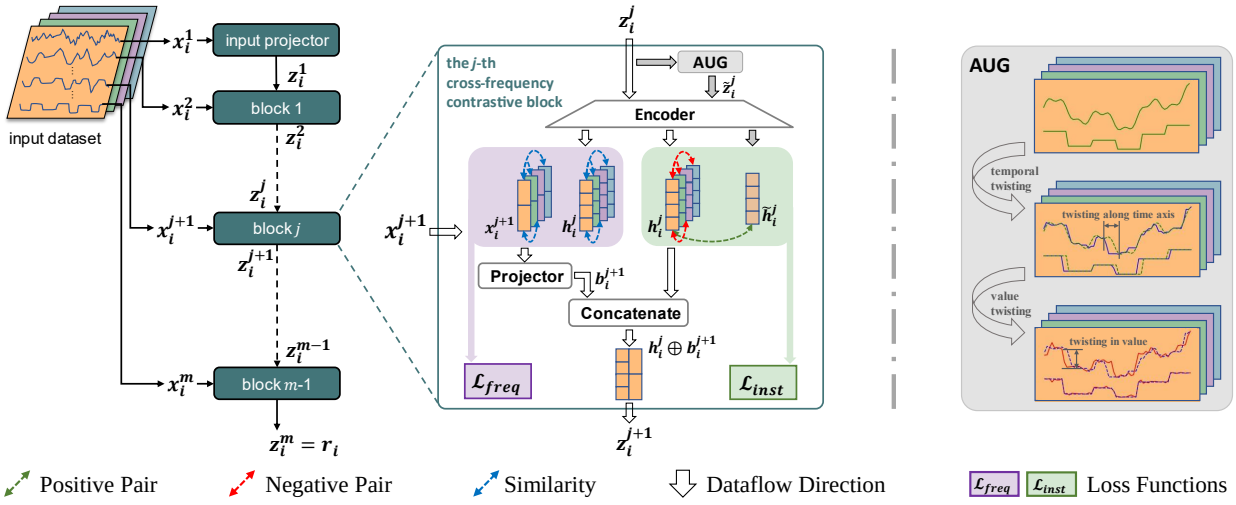


Figure 1. The architecture of MF-CLR. It hierarchically learns the representation from the highest frequency to the lowest using cross-frequency contrastive block. In each block, two groups of subseries with consecutive frequencies are processed.

these real-world challenges necessitates a self-supervised representation learning method capable of handling multiple frequencies inputs.

Unfortunately, most of the existing contrastive representation learning methods do not pay special attention to the modeling of multi-frequency time series. Consequently, their proposals in data augmentations, positive and negative pair selections, and contrastive losses may not be sufficiently suitable. For example, *permutation* is often used as a strong augmentation and its effectiveness has been validated (Eldele et al., 2021). In the multi-frequency setting, however, this augmentation method would increase the frequency of the low-frequency sequences and break the original frequency structure. Another example is that in uni-frequency setting, *temporal consistency* is an useful method in choosing positive pairs (Tonekaboni et al., 2021). But when processing the multi-frequency input data, this method may generate false positive pairs when the low-frequency sequences contain change points.

To address the above challenges, we propose a simple but effective model named Multi-frequency Contrastive Learning Representation (MF-CLR), which specializes in self-supervised representation learning for multi-dimensional time series with various frequencies. MF-CLR employs a set of cross-frequency contrastive blocks to establish a hierarchical mechanism along the feature dimension. For each block, two groups of subseries with consecutive frequencies are processed. It firstly augments a positive view on the high-frequency subseries using our proposed data augmentation method, Dual Twister, before implementing the instance discrimination pretext task. Simultaneously, low-frequency information is incorporated into the representation of high-frequency through cross-frequency consistency. Based on the aforementioned two consistency, representation on these frequencies are learned and the results are recursively served as the input of the next block. With this setting, MF-CLR explicitly learns invariant representations from the highest frequency subseries to the lowest ones until the information on all the frequency levels has been extracted. Extensive experiments are conducted to prove its effectiveness. To deliver a comprehensive assessment of the representation, comparable studies are investigated on five downstream tasks, which are classification, anomaly detection, short- and long-term forecasting, and imputation. We also conduct two real-world case studies, *i.e.*, stock price forecasting and net income forecasting, to show the usability of MF-CLR in practice. Experimental evidence shows that MF-CLR can deliver a leading and stable performance across all the tasks.

The contributions of our work are summarized as follows:

(1) We propose MF-CLR, a novel contrastive learning representation method designed for multi-dimensional time series containing multiple frequencies. It learns the representations from the highest frequency to the lowest in a hierarchical manner.

(2) We propose a new data augmentation method, Dual Twister, which adds noise along both dimensions of an input time series. The method can generate positive views with similar semantic meanings while preserving the original frequency structure.

(3) To capture the invariance between adjacent frequencies more effectively, we introduce cross-frequency consistency and propose corresponding contrastive losses.

(4) Extensive experiments are conducted to validate effec-

tiveness of MF-CLR effectiveness. Experimental evidence demonstrates that MF-CLR delivers a leading and consistent performance across all the downstream tasks and requires less scale of the target datasets.

2. Related Works

To discuss the self-supervised representation learning for time series, we have to first deviate our attention to other domains where this type of method thrives. In NLP domains, methods have been extensively studied dating back to Word2Vec (Mikolov et al., 2013) and the early form GPT (Radford et al., 2018), while in CV domains, methods based on contrastive learning have boomed in recent years. Their proposals threw light on time series domains a lot. For example, the momentum encoder introduced by MoCo (He et al., 2020), the projection head introduced by SimCLR (Chen et al., 2020), and the predictive pretext task introduced by CPC (Oord et al., 2018) are all more or less absorbed by later works for self-supervised time series representation learning. However, obstacles occur when these approaches are migrated directly to the time series domain. The main reason is that most of the approaches in other domains implement augmentations and pretext tasks by leveraging inductive bias, *e.g.*, image colorization (Tian et al., 2020) and image rotation (Gidaris et al., 2018) in CV domains and the mask language model (Devlin et al., 2019) in NLP domains, which is not always suitable in modeling time series data.

Although self-supervised representation learning is less studied for time series compared with domains like CV or NLP, many approaches have been proposed in recent years. T-Loss (Franceschi et al., 2019), which is enlightened by Word2Vec, is one of the pioneers implementing contrastive learning on time series data. TS-TCC (Eldele et al., 2021) uses a pair of augmentations, *i.e.*, a weak one and a strong one, and tries to capture the invariance between them. CA-TCC (Eldele et al., 2023) extends TS-TCC into the semi-supervised setting by leveraging the pseudo labels produced by TS-TCC. Another method focusing on data augmentation is InfoTS (Luo et al., 2023). It implements meta-learning to automatically select the appropriate data augmentation strategy and uses it for later contrasting. Some approaches try to leverage temporal properties for contrastive learning. For example, CoST (Woo et al., 2022) and ACST (Hu et al., 2023) capture seasonality and trend in the latent space while TF-C (Zhang et al., 2022) learns the invariant representation between time and frequency domain. As discussed before, domain-specific expert knowledge plays an important role in time series problems. Many approaches (Mohsenvand et al., 2020; Tonekaboni et al., 2021; Sarkar & Etemad, 2022; Raghu et al., 2023) are directly inspired by domain-specific problems, while other approaches, such as ExpCLR (Nonnenmacher et al., 2022), intent to incorporate expert features into the contrastive learning architecture. There are also methods trying to give a universal solution for all time series problems. TS2Vec (Yue et al., 2022) learns how to distinguish the positive and negative samples based on contextual consistency. TimesURL (Liu & Chen, 2023) proposes that enroll time reconstruction as a joint target enables better segment- and instance-level representation. Besides contrastive learning, Transformer-based pre-trained models are also becoming popular recently. TimeGPT (Garza & Mergenthaler-Canseco, 2023) leverages the principles of transfer learning to apply the time series foundation model pre-trained on the largest collection of publicly available time series based on its channel-independence setting. The forecasting task in the target datasets demonstrates its leading performance.

However, all of the aforementioned approaches fail to pay special attention to multi-frequency data. The multi-frequency inputs are simply treated as uni-frequency-sampled multi-dimensional time series. This leads to the inconsistency of performance in the multi-frequency setting as in the uni-frequency setting on which these algorithms are originally designed.

3. Proposed Method

3.1. Preliminary

The goal of MF-CLR is to learn a good embedding $r_{i,t} = f(x_{i,t}|\theta)$ from the training set $X = \{x_i|i \in 1, 2, ..., N\}$, where $r_{i,t} \in \mathbb{R}^L$, $x_i \in \mathbb{R}^{T \times D}$, T is the length, D is the input channel size and L is the dimension of the representation in the latent space. For each x_i there are m different frequencies across the total dimension of D, and we have $x = x_i^1 \oplus x_i^2 \oplus \cdots \oplus x_i^m$ dividing the total dimension into m groups of subseries, where each $x_i^j \in \mathbb{R}^{T \times d_j}$ and $\sum_{j=1}^{m} d_j = D$. For each pair of consecutive groups x_i^j and x_i^{j+1}, we define the frequency of x_i^j is higher than the frequency of x_i^{j+1}.

3.2. Architecture

The architecture of MF-CLR is illustrated in Figure 1.

Instead of training one deeper encoder, MF-CLR deploys a smaller and more easy-to-train encoder for each frequency level. The learning is in a hierarchical manner from the highest frequency to the lowest by the cross-frequency contrastive blocks, where two groups of subseries with consecutive frequencies are processed. For the j-th block, we first generate $\widetilde{z}_i^j$ using our proposed data augmentation strategy to create a twisted view on the block input z_i^j. Then both of the two views are encoded by the backbone encoder, which optimizes with the instance-wise contrasting between h_i^j

and $\widetilde{h}_i^j$, and the cross-frequency contrasting between h_i^j and x_i^{j+1}. To get the output of this block, we concatenate the latent vector h_i^j with the embedded lower-frequency input b_i^{j+1} along the feature dimension. Note that instead of using x_i^{j+1} directly, we use a non-linear projection to align the two consecutive frequency. Till this end, the output z_i^{j+1} contains the learned invariance of both the lower-frequency subseries' and the higher ones', and can be recursively served as the input for the $(j+1)$-th block.

3.3. Dual Twister

Data augmentation is an essential process in contrastive learning where positive samples are generated for later discrimination in solving the pretext task. A good data augmentation strategy adds deviation to create different views without breaking the semantic similarity between the augmented sequence and the input.

Due to the local time shifting, two similar time series will not always perfectly pair on each corresponding time step (Berndt & Clifford, 1994). Enlightened by this phenomenon, we present Dual Twister, which augments the input sequence in both temporal-wise and value. First, we randomly generate the alignment policy, pairing time steps between the input and the augmented sequence. This policy is shared across different frequency levels to keep the frequency structure unchanged. Following the alignment policy, the augmented sequence is twisted along the temporal horizon. Next, we set a total deviation value and assign it to each pair of time steps. This allows the augmented sequence to be twisted *w.r.t.* the value corresponding to each time step. Here the total deviation follows Proposition 3.1 and it ensures the second twist is not too large to break the semantic similarities. Due to the space limitation, we place the detailed process of implementing Dual Twister in Appendix A.2

Proposition 3.1. *The total deviation is the tight upper bound of the DTW distance between the augmented sequence and the original input.*

Using Dual Twister, we can generate a twisted view on the input subseries with a constraint of total deviation. The generated positive sample pairs share similar yet practical semantic meanings while maintaining the original frequency structure.

3.4. Cross-frequency Consistency

Aiming to capture the invariant representation between consecutive frequencies for each block, we come up with cross-frequency consistency which motivates the model with the following two properties:

Property 1: For higher-frequency subseries, the representation should follow the instance-wise consistency.

Property 2: The more similar the lower-frequency subseries are, the closer the representation of the higher-frequency subseries should be in the latent space.

The first property encourages the representation to capture the invariance inside the same frequency by implementing the instance discrimination task. As for the positive and negative sample selection strategy, the representations of the original subseries and the augmented subseries are considered positive, while the representations of different time series inside a batch are taken as negatives. Some works (Yue et al., 2022) argue that adding temporal contrast is necessary to capture invariance along the time dimension. However, since we augment the time series as a whole instead of adding noise on each time step, it is more reasonable to discriminate the whole instance of the time series. We show that introducing temporal consistency to MF-CLR brings no benefit in Section 4.6.1.

The second property aims to add lower-frequency information to the higher-frequency representations. By pulling the representations closer in the latent space based on the distance of the corresponding lower-frequency in the original space of the raw signals, the representations combine the invariance of both frequency levels to form a comprehensive result.

3.5. Contrastive Loss

For each frequency level, the contrastive loss is built up by two parts, *i.e.*, instance-wise contrastive loss and cross-frequency contrastive loss. Follow the idea of **Property 1** in Section 3.4, the instance-wise contrastive loss is formulized in Equation (1), where h and h' are representations encoded from the block input and the augmentations, respectively and B is the batch size.

$$\ell_{inst}^{i,t,j} = -\log \frac{\exp(h_{i,t}^j \cdot \tilde{h}_{i,t}^j)}{\sum_{k=1}^{B} \left(\exp(h_{i,t}^j \cdot \tilde{h}_{k,t}^j) + \mathbb{I}_{i \neq k} \exp(h_{i,t}^j \cdot h_{k,t}^j) \right)} \quad (1)$$

For the cross-frequency contrastive loss introduced by the **Property 2**, we apply a variation of the quadratic contrastive loss (Nonnenmacher et al., 2022), as shown in Equation (2).

$$\ell_{freq}^{i,t,j} = \sum_{k=1}^{B} \left(\frac{\|x_{i,t}^{j+1} - x_{k,t}^{j+1}\|_2}{\max_{l,n} \|x_{l,t}^{j+1} - x_{n,t}^{j+1}\|_2} - \|h_{i,t}^j - h_{k,t}^j\|_2 \right)^2 \quad (2)$$

The total loss is the weighted average between the two loss term controlled by the hyper-parameter α.

$$\mathcal{L}_{dual}^j = \sum_{i=1}^{N} \sum_{t=1}^{T} \left((1-\alpha) \cdot \ell_{inst}^{i,t,j} + \alpha \cdot \ell_{freq}^{i,t,j} \right) \quad (3)$$

3.6. Training

The hierarchical training from the highest frequency to the lowest can be implemented based on Algorithm 1. For each

block, we first generate an augmented view from the input by Dual Twister before encoding them separately by the backbone encoder. Loss for this frequency level is calculated based on Equation (3), and the parameters of this level's encoder are updated accordingly. Finally, we concatenate the encoded vector with projected lower-frequency subseries to get the block output, which also serves as the next block's input recursively.

Algorithm 1 Hierarchical training.

Initialize θ_1 to θ_{m-1}.
$e \leftarrow 0$
while $e < Epoch$ **do**
  for $j = 1$ **to** $m - 1$ **do**
    $\tilde{z}^j \leftarrow augment(z^j)$
    $h^j \leftarrow f(z^j|\theta_j)$
    $\tilde{h}^j \leftarrow f(\tilde{z}^j|\theta_j)$
    $loss_j \leftarrow \mathcal{L}^j_{dual}(h^j, \tilde{h}^j, x^{j+1})$
    $\theta_j \leftarrow \theta_j - \nabla loss_j(\theta_j)$
    $z^{j+1} \leftarrow h^j \oplus projector(x^{j+1})$
  end for
  $e \leftarrow e + 1$
end while

4. Experiments

To evaluate MF-CLR's usability, we perform extensive experiments on five downstream tasks. We enroll several self-supervised representation learning methods as baselines, including T-Loss (Franceschi et al., 2019), TS-TCC (Eldele et al., 2021), TNC (Tonekaboni et al., 2021), CPC (Oord et al., 2018), TS2Vec (Yue et al., 2022), CoST (Woo et al., 2022) and TF-C (Zhang et al., 2022). For comprehensiveness, some of the supervised methods are also enrolled, including DeepAR (Salinas et al., 2020), TCN (Bai et al., 2018), Informer (Zhou et al., 2021), Autoformer (Wu et al., 2021), FEDformer (Zhou et al., 2022), DLinear (Zeng et al., 2023), PatchTST (Nie et al., 2022) and TimesNet (Wu et al., 2023). Baselines are listed and introduced in Appendix E.

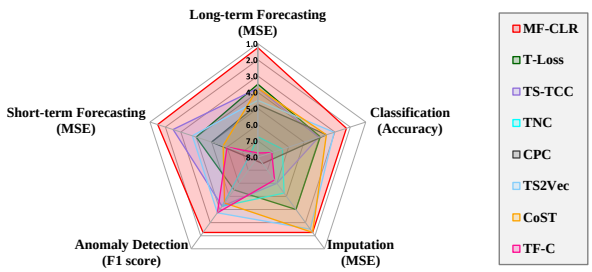


Figure 2. Average ranks on five downstream tasks compared with other self-supervised baselines.

Table 1. Performance for classification on 30 UEA datasets. Metrics are macro averaged.

Methods	Acc.	P.	R.	F1	ROC	PRC
MF-CLR	0.6487	**0.6411**	0.6297	0.6130	**0.7989**	**0.6672**
self-supervised						
T-Loss	0.6044	0.6094	0.5855	0.5687	0.7876	0.6496
TS-TCC	0.5892	0.5884	0.5642	0.5465	0.7828	0.6080
TNC	0.2579	0.1836	0.2363	0.1791	0.4989	0.2470
CPC	0.6179	0.6063	0.5987	0.5814	0.7954	0.6492
TS2Vec	0.6167	0.6006	0.5957	0.5751	0.7821	0.6424
CoST	0.6262	0.6139	0.6043	0.5848	0.7738	0.6282
TF-C	0.2577	0.2113	0.2444	0.2048	0.5087	0.2563
supervised						
DLinear	0.6207	0.6134	0.6085	0.5999	0.7841	0.6407
PatchTST	**0.6540**	0.6395	**0.6364**	**0.6191**	0.7978	0.6598
TimesNet	0.6436	0.6395	0.6271	0.6085	0.7980	0.6633
Autoformer	0.5902	0.5865	0.5706	0.5549	0.7558	0.5917
Informer	0.6468	0.6396	0.6296	0.6008	0.7869	0.6453
FEDformer	0.6196	0.6035	0.6028	0.5819	0.7876	0.6393

4.1. Data

Most of the datasets that are heavily used in the time series domain are sampled in uni-frequency, leaving them inappropriate for direct usage. To balance between the general acceptance and suitability, we resample these public datasets by different frequencies along the feature dimension to meet the multi-frequency setting. Details about the public datasets preprocessing can be found in Appendix B.

4.2. Classification

The performance of classification is evaluated on 30 UEA datasets. For all the self-supervised methods, we train an SVM with RBF kernel on the learned representation of the whole sequence. Table 1 summarizes the average performance across all the datasets. The best performance ones are bolded and the second best ones are underlined.

Compared with self-supervised methods, MF-CLR outperforms all the baselines in all the six metrics. When the comparison is made against supervised methods, MF-CLR realizes the best performance *w.r.t.* three metrics and the second best *w.r.t.* the other three ones.

4.3. Anomaly Detection

Anomaly detection is tested on four public datasets, *i.e.*, SMD, SMAP, MSL and SWaT. For self-supervised methods, a ridge regressor is firstly trained on the learned representations r_t to forecast the observation y_t. Then we list all the MSE between $\hat{y}_t$ and y_t, and determine a threshold to separate between the normal and the anomalous. Results summarized in Table 2 illustrate that MF-CLR achieve the best performance *w.r.t.* accuracy, precision and F1 score, outperforming the second for 0.63%, 1.69% and 2.22%, respectively. The full results can be found in Appendix F.2.

Table 2. Performance for anomaly detection. For datasets containing multiple subsets, the metrics are averaged across the datasets.

Methods	Acc.	P.	R.	F1
MF-CLR	**0.9689**	**0.6203**	0.8036	**0.6798**
self-supervised				
T-Loss	0.9498	0.4922	0.5696	0.5194
TS-TCC	0.9623	0.6034	0.7707	0.6400
TNC	0.9626	0.5817	0.7930	0.6439
CPC	0.9259	0.3815	0.5686	0.4334
TS2Vec	0.9591	0.5827	**0.8499**	0.6576
CoST	0.9582	0.5662	0.8013	0.6318
TF-C	0.9612	0.5991	0.8006	0.6475
supervised				
DLinear	0.9596	0.5960	0.5645	0.5703
PatchTST	0.9613	0.5923	0.5665	0.5685
TimesNet	0.9574	0.5648	0.5384	0.5409
Autoformer	0.9522	0.5839	0.4672	0.5056
Informer	0.9511	0.5759	0.4590	0.4984
FEDformer	0.9520	0.5794	0.4651	0.5024

Table 3. Performance for short-term and long-term forecasting.

Methods	traffic	ETTm1	ETTm2	weather
short-term forecasting				
MF-CLR	0.7033	**0.0479**	**0.1227**	0.1635
T-Loss	0.7179	0.0487	0.1310	0.4094
TS-TCC	**0.6206**	0.0574	0.1331	**0.1314**
TNC	1.6992	0.0971	0.1665	0.6142
CPC	0.9947	0.0483	0.3653	0.1874
TS2Vec	0.8800	0.0601	0.1264	0.2171
CoST	0.9604	0.0699	0.1436	0.3583
TF-C	0.9218	0.1914	0.1509	0.2934
long-term forecasting				
MF-CLR	**0.7466**	**0.0495**	**0.2060**	0.5496
T-Loss	0.9546	0.0712	0.2163	0.7149
TS-TCC	1.1737	0.0503	0.4257	0.6202
TNC	2.0632	0.0938	0.5754	1.0166
CPC	1.6865	0.1059	0.4915	**0.4811**
TS2Vec	0.8199	0.1073	0.3597	0.7656
CoST	0.9886	0.0648	0.2119	0.7979
TF-C	2.1745	0.5142	0.8250	1.0066

4.4. Short- and Long-term Forecasting

The performance of forecasting is evaluated on traffic, ETTm1, ETTm2, and weather. The forecasting is made on different frequencies and horizons. For short-term forecasting, we set the forecast horizon to be less than or equal to a week, while for long-term forecasting the horizon is one week above. For all the self-supervised methods, we train a ridge regressor on the learned representations to get the forecast results. Table 3 summarizes the average performance across the four datasets *w.r.t.* these two tasks and the metric used is MSE. We place the detailed results in Appendix F.1 due to the space limitation. From Table 3 we can conclude that MF-CLR achieves the best performance in both short- and long-term forecasting.

4.5. Imputation

We select the same four datasets used in Section 4.4 to evaluate the performance of imputation. Different from forecasting, imputation can use both historical observations and forward values to predict the masked values in the middle. For the self-supervised methods, we train two ridge regressors with one predicting the masked timestamp forward while the other one giving predictions backward. The final output is the averaged results given by these two regressors. To deliver a robust and comprehensive assessment, we follow TimesNet (Wu et al., 2023) to mask these datasets by 12.5%, 25.0%, 37.5% and 50%. The metric shown in Table 4 is averaged MSE across the 4 mask ratios. Full results are shown in Appendix F.3.

Results show that MF-CLR leads in most datasets except traffic when compared with self-supervised methods and leads in traffic and weather when compared with supervised methods. Since temporal contrasting is not enrolled based on our data augmentation, our method intentionally focuses more on less refined granularity. On the other hand, supervised end-to-end training methods can learn better neighborhood relationships by modeling each time step, like the attention mechanism used by Informer. This explains why MF-CLR fails to achieve the best performance when compared with supervised baselines.

4.6. Analysis

4.6.1. Ablation Study

In order to verify the effectiveness of each component in MF-CLR, we demonstrate comparisons between the full MF-CLR and different model variants. These variants are (1) w/o Dual Twister, (2) w/o cross-frequency contrast, (3) w/o low-frequency projector, (4) w/o hierarchical contrast. We use ETTm2 with hourly OT as the benchmark and the average performance gaps are summarized in Table 5. Details about ablation settings can be found in Appendix C.2.

As discussed in Section 3.4, the temporal contrast is theoretically not effective for MF-CLR. To give experimental evidence of this proposal, we add temporal contrastive loss term, as shown in Equation (4), and use Equation (5) to re-train MF-CLR. The result in Table 5 shows enrolling temporal contrast drags the performance significantly.

$$\ell_{temp}^{i,t,j} = -\log \frac{\exp(h_{i,t}^{j} \cdot \tilde{h}_{i,t}^{j})}{\sum_{t \in \mathbb{W}} \left(\exp(h_{i,t}^{j} \cdot \tilde{h}_{i,t'}^{j}) + \mathbb{I}_{t \neq t'} \exp(h_{i,t}^{j} \cdot h_{i,t'}^{j}) \right)} \quad (4)$$

Table 4. Performance for imputation.

Methods	traffic	ETTm1	ETTm2	weather
comparison against self-supervised methods				
MF-CLR	0.4844	**0.0852**	**0.0400**	**0.0159**
T-Loss	0.4632	0.1233	0.0485	0.0323
TS-TCC	0.4600	0.1349	0.0687	0.0556
TNC	0.4447	0.1246	0.0677	0.0573
CPC	0.7128	0.1387	0.0661	0.0963
TS2Vec	**0.4025**	0.0864	0.0483	0.0363
CoST	0.4072	0.0969	0.0424	0.0274
TF-C	0.4991	0.1261	0.0646	0.0586
comparison against supervised methods				
MF-CLR	0.4844	0.0852	0.0400	**0.0159**
DLinear	**0.2444**	0.0777	0.0364	0.0240
PatchTST	0.9496	0.6657	0.1074	0.0968
TimesNet	4.0906	0.4303	0.1264	0.1883
Autoformer	0.5511	0.1293	0.0466	0.0413
Informer	0.5794	**0.0399**	**0.0187**	0.0183
FEDformer	0.5378	0.0543	0.0315	0.0218

$$\mathcal{L}_{tri}^{j} = \sum_{i=1}^{N} \sum_{t=1}^{T} \left((1-\alpha) \cdot (\ell_{inst}^{i,t,j} + \ell_{temp}^{i,t,j}) + \alpha \cdot \ell_{freq}^{i,t,j} \right) \quad (5)$$

4.6.2. Encoder Selection

We first replace the encoder in MF-CLR by MLPs, RNN and Transformer (Vaswani et al., 2017) with similar parameter scale and results are summarized in Table 5.

Next, we show that our parameter chosen for the TCN encoder is reasonable. Figure 3 illustrates the relationship between the parameter scale and the MSE on the ETTm2 with three different OT frequencies. It shows that the parameter scale we choose balances between efficiency and performance.

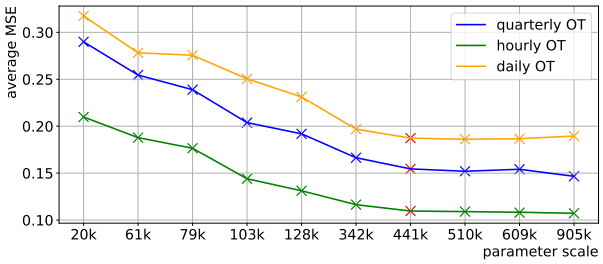


Figure 3. Relationship between the parameter scale and the MSE on ETTm2 for the forecasting task. The red cross represents the parameter scale corresponding to our chosen encoder. The performance drops significantly when the model capacity is lowered, while a deeper encoder brings little improvement in performance.

Table 5. Analysis on ETTm2 with hourly OT for forecasting.

Model Variants	MSE	MAE
full MF-CLR	0.110	0.252
Ablation		
w/o Dual Twister	+ 0.080	+ 0.049
w/o cross-frequency contrast	+ 0.107	+ 0.077
w/o low-frequency projector	+ 0.095	+ 0.036
w/o hierarchical contrast		
→ in QH-D setting	+ 0.011	+ 0.015
→ in Q-HD setting	+ 0.056	+ 0.037
Encoder Selection		
MLPs	+ 0.077	+ 0.063
LSTM	+ 0.081	+ 0.054
Transformer	+ 0.014	+ 0.008
Temporal Consistency		
w/ temporal contrast	+ 0.102	+ 0.152

4.6.3. Visualization

This section gives the visualized explanation of MF-CLR's effectiveness. Figure 5a visualizes representation of ERing from UEA using t-SNE (Van der Maaten & Hinton, 2008). It illustrates clear boundaries among clusters and few wrongly allocated points. Figure 5b shows the daily OT of ETTm1 and the learned representations, which have various frequencies and sharp change points corresponding to the daily OT. In Figure 4c, we select E-7 from the SMAP dataset and visualize its telemetry value, and the top 64 representation dimensions with the largest variances. It shows the learned representations on disturbance demonstrate significantly different patterns, indicating MF-CLR's effectiveness in the downstream anomaly detection task. Figure 4d corresponds to the weather dataset for imputation. We can see the missing values in the representations are filled, demonstrating MF-CLR's ability in extracting local information and depicting the adjacent relationship.

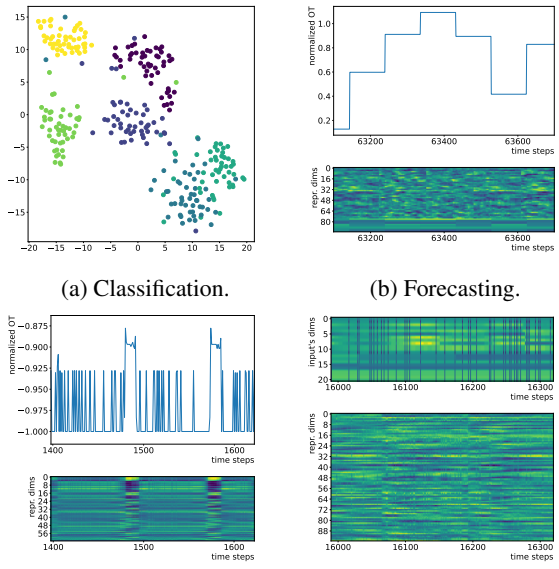
4.6.4. Transfer Learning

One of the advantages of self-supervised learning is the consistent performance delivered by a pre-trained model on small datasets based on the idea of transfer learning. Here we dive into the imputation tasks to show MF-CLR's performance under this setting. For ETTm1 → ETTm2 (ETTm2 → ETTm1), MF-CLR is pre-trained on ETTm1 (ETTm2) and fine-tuned on ETTm2 (ETTm1). These two datasets share similar semantic meanings, so they are suitable for the transfer learning experiment. To evaluate the performance on different target dataset scales, we also reduce the size of target datasets to 75%, 50%, 37.5% and 25% of the original scale, respectively.

Figure 5 illustrates that MF-CLR can deliver a much more

Table 6. Results of two real-world problems. We use MAPE (%), a well-accepted metric in financial domain, to measure the performance.

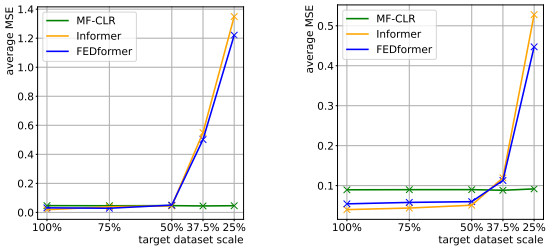
Experimental Scenarios			MF-CLR	T-Loss	TS-TCC	TNC	CPC	TS2Vec	CoST	TF-C	DeepAR	TCN
stock price	US	stable market	**11.71**	24.39	13.25	25.41	15.76	13.78	12.63	16.43	12.10	11.75
		volatile market	**16.44**	29.66	17.02	26.18	24.30	17.08	20.11	19.76	20.32	27.24
	CN	stable market	14.50	19.91	18.22	22.03	19.56	16.49	16.37	25.48	**13.17**	21.16
		volatile market	16.27	40.01	15.08	17.77	18.05	**14.60**	17.99	20.90	15.22	20.55
net income	US	stable market	**28.71**	48.55	35.82	38.19	40.28	39.41	35.77	41.87	38.91	36.80
		volatile market	**49.32**	80.22	59.00	79.07	77.34	60.75	56.61	61.89	74.39	76.85
	CN	stable market	21.64	35.13	24.36	36.21	34.05	22.74	25.50	33.22	30.63	**20.21**
		volatile market	**33.85**	51.22	40.11	47.05	46.83	33.94	39.76	42.88	56.80	43.14



(a) Classification. (b) Forecasting.

(c) Anomaly detection. (d) Imputation.

Figure 4. Visualization of various downstream tasks.



(a) ETTm1 → ETTm2. (b) ETTm2 → ETTm1.

Figure 5. Transfer learning for the imputation task.

stable performance across different target dataset scales compared with the top-performanced supervised methods. Besides, the performance of TS2Vec and FEDformer drops significantly after the target datasets are cut below 37.5%. This finding supports the idea that MF-CLR requires less data volume for downstream tasks compared with supervised methods.

4.7. Real-world Case Studies

Finally, we demonstrate the effectiveness of MF-CLR in practice by implementing two real-world case studies. The first one is stock price forecasting, aiming to forecast the close price of the next month's trading days. The second one is net income forecasting, aiming to forecast the net income of the next report period. These two tasks can demonstrate the performance of forecasting both high-frequency target (stock price) and low-frequency target (net income). For comprehensiveness, we randomly pick 25 stocks listed in the US market and 25 stocks listed in the Chinese market, and the tests are made on both periods of stable market and volatile market. Details about the test data can be found in Appendix D. For all the self-supervised methods, we follow Section 4.4 to get the forecast results.

Results are summarized in Table 6. Of all the 8 subtasks, MF-CLR achieves the top two performance in 7. In terms of the average performance on stock price forecasting, MF-CLR outperforms the second best method (TS-TCC) for 1.06% *w.r.t.* US market and achieves second by underperforming DeepAR for 1.19% *w.r.t.* Chinese market. For the average performance on net income forecasting, MF-CLR outperforms the second best method (CoST) for 7.18% *w.r.t.* US market and outperforms the second best method (TS2Vec) for 0.60% *w.r.t.* Chinese market.

5. Conclusion and Future Work

This paper presents MF-CLR, a self-supervised representation learning method designed for multi-frequency time series. The work shows our specifically designed data augmentation, cross-frequency consistency as well as hierarchical contrastive block contribute to MF-CLR's performance enhancement compared with previous works. Extensive

experiments *w.r.t.* five downstream tasks and two real-world case studies show that MF-CLR can deliver a leading yet consistent performance in the multi-frequency scenario.

Next, we will follow the idea of TimeGPT and pre-train MF-CLR on the largest collection of real-world time series datasets. Unlike TimeGPT which treats the multi-frequency training data as uni-frequency-sampled time series, our proposed data augmentation, cross-frequency consistency and contrastive loss function enable MF-CLR to better process multi-frequency time series. Considering that different real-world datasets may be resampled in different frequencies, MF-CLR is more suitable for forming a general pre-trained model. We think this future work will contribute more to the time series domain.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

Berndt, D. J. and Clifford, J. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining*, pp. 359–370, 1994.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:52967399.

Eldele, E., Ragab, M., Chen, Z., Wu, M., Kwoh, C. K., Li, X., and Guan, C. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2352–2359, 2021.

Eldele, E., Ragab, M., Chen, Z., Wu, M., Kwoh, C.-K., Li, X., and Guan, C. Self-supervised contrastive representation learning for semi-supervised time-series classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32, 2019.

Gao, T., Yao, X., and Chen, D. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Garza, A. and Mergenthaler-Canseco, M. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1v4N2l0-.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Houssein, E. H., Dirar, M., Abualigah, L., and Mohamed, W. M. An efficient equilibrium optimizer with support vector regression for stock market prediction. *Neural Computing and Applications*, pp. 1–36, 2022.

Hu, J., Hu, Z., Li, T., and Du, S. A contrastive learning based universal representation for time series forecasting. *Information Sciences*, 635:86–98, 2023.

Liu, J. and Chen, S. Timesurl: Self-supervised contrastive learning for universal time series representation learning. *arXiv preprint arXiv:2312.15709*, 2023.

Luo, D., Cheng, W., Wang, Y., Xu, D., Ni, J., Yu, W., Zhang, X., Liu, Y., Chen, Y., Chen, H., et al. Time series contrastive learning with information-aware augmentations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4534–4542, 2023.

Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*,

2013. URL https://api.semanticscholar.org/CorpusID:5959482.

Mohsenvand, M. N., Izadi, M. R., and Maes, P. Contrastive representation learning for electroencephalogram classification. In *Machine Learning for Health*, pp. 238–253. PMLR, 2020.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. *ArXiv*, abs/2211.14730, 2022. URL https://api.semanticscholar.org/CorpusID:254044221.

Nonnenmacher, M. T., Oldenburg, L., Steinwart, I., and Reeb, D. Utilizing expert features for contrastive learning of time-series representations. In *International Conference on Machine Learning*, pp. 16969–16989. PMLR, 2022.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Punia, S. and Shankar, S. Predictive analytics for demand forecasting: A deep learning-based decision support system. *Knowledge-Based Systems*, 258:109956, 2022.

Punia, S., Singh, S. P., and Madaan, J. K. A cross-temporal hierarchical framework and deep learning for supply chain forecasting. *Computers & Industrial Engineering*, 149:106796, 2020.

Qin, Y., van der Schaar, M., and Lee, C. T-phenotype: Discovering phenotypes of predictive temporal patterns in disease progression. In *International Conference on Artificial Intelligence and Statistics*, pp. 3466–3492. PMLR, 2023.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Raghu, A., Chandak, P., Alam, R., Guttag, J., and Stultz, C. Sequential multi-dimensional self-supervised learning for clinical time series. In *International Conference on Machine Learning*, pp. 28531–28548. PMLR, 2023.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

Sarkar, P. and Etemad, A. Self-supervised ecg representation learning for emotion recognition. *IEEE Transactions on Affective Computing*, 13(3):1541–1554, 2022. doi: 10.1109/TAFFC.2020.3014842.

Saveliev, E. S. and van der Schaar, M. Temporai: Facilitating machine learning innovation in time domain tasks for medicine. *arXiv preprint arXiv:2301.12260*, 2023.

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M. (eds.), *Computer Vision – ECCV 2020*, pp. 776–794, Cham, 2020. Springer International Publishing.

Tonekaboni, S., Eytan, D., and Goldenberg, A. Unsupervised representation learning for time series with temporal neighborhood coding. *arXiv preprint arXiv:2106.00750*, 2021.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PilZY3omXV2.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., and Xu, B. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8980–8987, 2022.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

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

Zhao, L., Kong, S., and Shen, Y. Doubleadapt: A meta-learning approach to incremental learning for stock trend forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3492–3503, 2023.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pp. 27268–27286. PMLR, 2022.

A. Data Augmentation

A.1. Summary of the Most Used Methods

Data augmentation is an essential process in contrastive learning where two views are generated from the input data. A good data augmentation strategy should introduce noise to separate the two views while maintaining the essential invariance between the two. Methods most used by previous works are summarized as follows:

- scaling: scale (multiply) the whole time series by a random scaler value.
- shifting: shift (add) the whole time series by a random scalar value.
- jittering: add noise, *e.g.*, i.i.d. Gaussian noise, to each time step.
- permutation: cut the whole time series into several subseries and permute them in random orders.
- random cropping: crop two overlapping subseries to create a left and a right view.
- random masking: randomly mask some time steps.

Visualized examples of the above methods on a synthetic input data are shown in Figure 6.

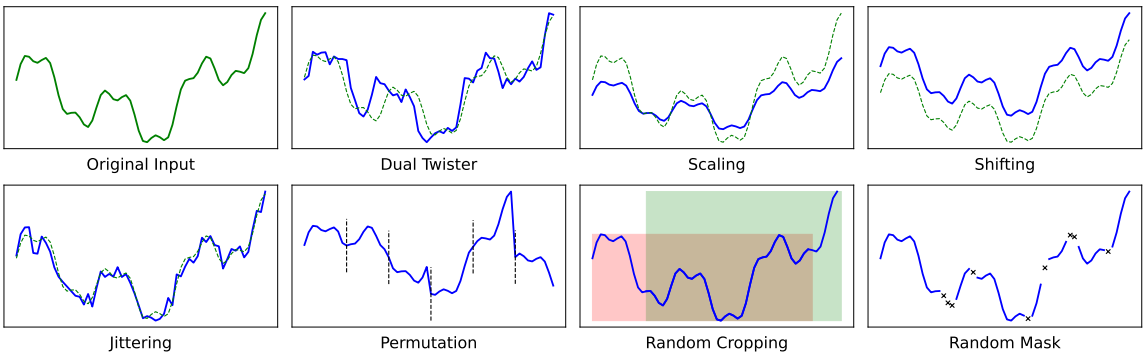


Figure 6. Visualization of our proposed Dual Twister and the data augmentations most used by previous works.

A.2. Dual Twister

Our proposed augmentation strategy can be used to augment the given original subsequence with arbitrary lengths and ensure that the deviation of the generated augmentation sequence from the original sequence is within a controllable range. In a sense, Dual Twister can be seen as an inverse process of DTW. DTW finds the path first using dynamic programming before calculating the distance along this path, whereas Dual Twister generates the alignment policy first before assigning the total deviation to get the augmented subsequence. However, DTW has a complexity of $O(N^2)$ while Dual Twister's is $O(N)$ for finding the path (alignment policy).

First, determine the subsequence that requires data augmentation, and at this point, the length of the augmentation subsequence is also uniquely determined. Then, the probabilistic alignment policy between the original subsequence and the augmented subsequence is generated strictly in chronological order to determine the matching relationship between the augmentation of each time step and the original subsequence. The probability of each alignment choice is calculated based on the Equation (6),

$$\begin{cases} dist(i,j) = (i-j)^2 \\ \Delta dist(i,j) = ((i+\Delta i) - (j+\Delta j))^2 \\ Prob(\Delta i, \Delta j) = \exp\left(-\frac{dist(i,j)}{10} \times \Delta dist(i,j)\right) / \sum_{k,l} (\exp\left(-\frac{dist(k,l)}{10}\right) \times \Delta dist(k,l)) \\ s.t. \quad (\Delta i, \Delta j) \in (1,0), (0,1), (1,1) \end{cases} \tag{6}$$

, where i is the index of the original subsequence, j is the index of the augmented subsequence and $dist(i, j)$ represents the current lateral offset. $Prob(\Delta i, \Delta j)$ is used to determine the next alignment choice, which is based on the following policy.

If the current lateral offset is small, the next step is inclined to select the direction that increases the lateral shift, otherwise, the next step is inclined to decrease the shift.

In doing so, the generated augmentation subsequence can simultaneously satisfy randomness and ensure the similarity of data distribution.

After the alignment is generated, we then set the overall offset value to constrain the deviation between the augmented and the original subsequence. The overall offset should be randomly allocated for each matching pair according to the determined alignment policy. If there is a one-to-many match between the augmented subsequence and the original subsequence at a specific time step, it is necessary to ensure that the deviation is assigned on a pro-rata basis.

A.3. Proof of Proposition 3.1

Denote t as the temporal horizon, T as the current time step, D as the overall deviation we set, D_i as the accumulated total deviation (calculated by DTW) until ith time step, and d_i as the deviation between the matching pair in ith time step.

Given the original and the augmented time series, when the time step $T = 1$, it is obvious that the given total deviation value D is the upper bound of the distance between them.

Assume when $T = t - 1$, we have

$$D_T = D_{t-1} \leq D - d_t \tag{7}$$

Next, when $T = t$, the accumulated deviation satisfies

$$D_T = D_t = D_{t-1} + d_t \leq D - d_t + d_t = D \tag{8}$$

Thus, Proposition 3.1 is proven.

B. Public Datasets Preprocessing

B.1. Anomaly Detection

The original SMAP dataset has 55 channels with 25 dimensions for each and the original MSL dataset has 27 channels with 55 dimensions for each. Both of them are sampled every 1 minute, and only the telemetry value is continuous while other dimensions are either 1 or 0, representing whether the commands have been successfully sent. In our experiments, we keep the telemetry value in its original frequency and resample other dimensions into quarterly and hourly. For SMAP, d_2 to d_{13} are resampled into quarterly and d_{14} to d_{25} are resampled into daily. For MSL, d_2 to d_{28} are resampled into quarterly and d_{29} to d_{55} are resampled into daily.

The original SMD dataset is also sampled every 1 minute, and it contains 28 entities with each of them having 38 dimensions of data. The original SWaT dataset is sampled every 1 second and the dimension is 51. For SMD, we keep d_1 to d_{13} in their original frequency, and the d_14 to d_{26} and d_27 to d_{38} are resampled into quarterly and hourly, respectively. For SWaT, we keep d_1 to d_{25} in their original frequency, and the d_26 to d_{34} and d_35 to d_{51} are resampled every minute and every quarter, respectively.

B.2. Forecasting and Imputation

The original traffic dataset has 862 features including the forecast target, *i.e.*, OT and 17544 timesteps sampled hourly. We retain the former 400 features unchanged and resample the latter 461 features to daily, *i.e.*, sum up every 24 values. Besides, the OT is also processed in two settings, retaining hourly or resampling into daily.

Both ETTm1 and ETTm2 originally have 7 features including OT and 69680 timesteps sampled quarterly. We resample the features into three granularities. The HUFL and HULL are retained as quarterly observations. The MUFL and MULL are resampled hourly while LUFL and LULL are resampled daily. Also, the OT is resampled in all three frequencies.

The original weather dataset has 21 features including OT and 52696 timesteps sampled every 10 minutes. The dataset is resampled into four granularities. For p, T, Tpot and Tdew, we retain the original frequency. For rh, VPmax, VPact, VPdef,

sh, H2OC and rho, we resample them hourly by picking a value every six observations. The daily resampled data contains wv, max. wv and wd. Finally, we resample the rest features, *i.e.*, rain, raining SWDR, PAR, max. PAR and Tlog, into weekly observations. The OT is resampled in all four different frequencies as well.

The number of features, timesteps and granularity of both the original and the processed datasets are summarized in Table 7.

Table 7. Summary of the resampling on traffic, ETTm1, ETTm2 and weather.

	original			after preprocessing		
	features	timesteps	granularity	features	timesteps	granularity
traffic	862	17544	1 hour	862	17544	1 hour, 1 day
ETTm1	7	69680	15 min	7	69680	15 min, 1 hour, 1 day
ETTm2	7	69680	15 min	7	69680	15 min, 1 hour, 1 day
weather	21	52696	10 min	21	52696	10 min, 1 hour, 1 day, 1 week

B.3. Classification

The UEA is an integrated dataset consisting of 30 time series classification problems of various domains. In the preprocessing stage, we do our best to dig into the details of each dataset carefully in order to resample the features into the multi-frequency setting while maintaining physical meanings. For example, the "BasicMotion" dataset is generated from students performing four different activities, which are walking, resting, running, and playing badminton. The 6 features are collected by two sensors, a 3D accelerometer and a 3D gyroscope, with each of them providing values in the x-, y- and z-axis. The original data is sampled at 10Hz lasting for 10 seconds, which in total provides 100 timestamps of observations. When resampling, we keep the 3 dimensions of accelerometer data to be original and resample the 3 dimensions of gyroscope data every two timestamps. As a result, we generate a multi-frequency dataset in which the accelerations are sampled at 10Hz while the angular velocities are sampled at 5Hz.

The resampling strategies for all the 30 UEA datasets are summarized in Table 8. The notation d_i in the table represents the i-th feature (starting from 1).

C. Experimental Details

C.1. Parameter Setting

For all the downstream tasks, we set the batch size to be 32, the initial learning rate to be 1E-3 with 70% decay every 10 steps, and α to be 1E-5.

The encoder of MF-CLR contains 4 hidden dilated convolutional layers with feed-forward connections between consecutive blocks. The dilation is set to be 2^i for the i-th layer with the hidden channel size of 64 and the kernel size of 3. Also, a 10% dropout is added to enhance robustness.

The low-frequency projector is a two-layer-MLPs, where the input channel size is the dimension of the low-frequency subseries, the hidden dimension is 16 and the output dimension is 32.

The datasets are divided into training (70%), validation (10%) and test (20%) unless the given dataset has already been divided. The MF-CLR is trained on both the training set and the validation set, while the downstream model, *e.g.*, ridge regressor for forecasting and SVM for classification, is trained only on the training set. Hyper parameters of the downstream model are optimized by grid searching on the validation set. The encoding length of the input data is set to be 128 if it contains more than 2048 time steps, otherwise the encoding length is set to be the length of the input data.

C.2. Ablation

C.2.1. W/O DUAL TWISTER

For each frequency level, the $\hat{z}_i^j$ is no longer augmented from z_i^j by our proposed Dual Twister. In order to make the contrastive learning work successfully, we augment z_i^j by jittering. At each time step, Gaussian noise $\varepsilon_t \sim N(0, 0.5)$ is

added to $z_{i,t}^{j}$.

C.2.2. W/O CROSS-FREQUENCY CONTRAST

Instead of training the model based on Equation (3), we only use the instance-wise contrastive loss by setting the hyper-parameter α to be 0.

C.2.3. W/O LOW-FREQUENCY PROJECTOR

When generating each block's output, h^j is concatenated with x^{j+1} directly.

C.2.4. W/O HIERARCHICAL CONTRAST

The resampled ETTm2 dataset contains three different frequencies, *i.e.*, quarterly, hourly and daily. Instead of contrasting hierarchically with each block processing two subseries with consecutive frequencies, we divide the feature dimension into two frequency levels and only contrast one time. In the QH-D setting, the quarterly and hourly subseries are treated as low-frequency ones while the daily subseries are treated as high-frequency ones. In the Q-HD setting, the quarterly subseries are treated as low-frequency ones while the hourly and the daily subseries are treated as high-frequency ones.

D. Details about Real-world Case Studies

D.1. US Stock Data

We randomly choose 25 stocks which are listed publicly in the US stock market to form our dataset. The temporal horizon of the data we collected is from 2003-05-26 to 2023-05-25. For anyone who is listed after 2003-05-26, we fill the empty value by zero.

The features included are listed as follows.

(1) individual stock data: open price, close price, the intra-day highest price, the intra-day lowest price, quantity, volume, amplitude, change, percentage change, turnover.

(2) financial report: cash, account receivable, inventory, total current asset, long-term investment, PP&E (property, plant & equipment),equity investment, total fixed asset, total asset, account payable, current liability, long-term debt, fixed liability, total liability, total equity, revenue, COGS (cost of goods sold), R&D (research & development),operating expense, other income, net income, OCI(other comprehensive income), net CFO(cash flows from operating activities), net CFI(cash flows from investing activities), net CFF(cash flows from financing activities).

(3) global macro index: BDI (Baltic dry index), SOX (PHLX semiconductor sector).

(4) US macro index: GDP, core CPI, PMI, unemployment rate, industrial production, durable goods orders, factory orders, business inventory, NAHB house market index, FHFA house price index, new house starts, new house sales, personal spendings, retail sales index.

For stable market period, we forecast the 21 timesteps ahead (approximately one month of trading day) from 2019-10-15 *w.r.t.* stock price prediction, and 1 observation ahead from 2019-12-02 *w.r.t.* net income prediction.

For volatile market period, we forecast the 21 timesteps ahead from 2023-04-20 *w.r.t.* stock price prediction, and 1 observation ahead from 2022-12-01 *w.r.t.* net income prediction.

D.2. Chinese Stock Data

For stocks listed in Chinese stock exchange, we also randomly choose 25 stocks to form our dataset. The temporal horizon of the data we collected is from 2003-05-26 to 2023-05-25. For anyone who is listed after 2003-05-26, we fill the empty value by zero.

The features included are listed as follows.

(1) individual stock data: open price, close price, the intra-day highest price, the intra-day lowest price, quantity, volume, amplitude, change, percentage change, turnover.

(2) financial report: net fixed income, construction in progress, inventory, account receivable, total equity, hold-to-maturity investment, current liability, total current asset, total liability, cash, total asset, long-term equity investment, fixed liability, total fixed asset, operational revenue, OCI, operational return, operational cost, net return, net CFO, net CFI, net CFF.

(3) global macro index: BDI, SOX.

(4) Chinese macro index: currency rate, GDP, CPI, PPI, industry PMI, service PMI, interest rate (using LPR rate with terms of less than one year, one year, five years and above five years, respectively), FDI (foreign direct investment), total retail sales of consumer goods, total social financing, total import, total export, value added of industries, M2 growth, consumer confidence, residential leverage ratio, governmental leverage ratio, non-financial department leverage ratio, enterprise booming index.

For stable market period, we forecast the 21 timesteps ahead (approximately one month of trading day) from 2019-10-15 *w.r.t.* stock price prediction, and 1 observation ahead from 2019-12-02 *w.r.t.* net income prediction.

For volatile market period, we forecast the 21 timesteps ahead from 2023-04-20 *w.r.t.* stock price prediction, and 1 observation ahead from 2022-12-01 *w.r.t.* net income prediction.

E. Baselines

E.1. Self-supervised Methods

T-Loss: This is one of the pioneers in time series contrastive representation learning. It is enlightened by the succeed of Word2Vec in NLP domains and the contrasting is constructed using similar ideas of Word2Vec. The model uses subseries consistency to build up positive and negative sample pairs and use triplet loss to train the backbone encoder. We use the open source code from https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries in our experiments.

TS-TCC: This method implements two data augmentations, *i.e.*, a weak one and a strong one, to create two correlated views on the same subseries. Then, the cross-view prediction is implemented, using the past of one view to forecast the future of another. Further, the contextual contrasting maximizes the similarity among different contexts of the same sample while minimizing the similarity of different samples. We use the open source code from https://github.com/emadeldeen24/TS-TCC in our experiments.

TNC: The method assuming windows within a neighbourhood possess similar properties. The neighbourhood boundaries are determined automatically using the properties of the signal and statistical testing. Incorporating concepts from Positive Unlabelled Learning, the signals outside of this neighbourhood as unlabelled samples, are weight-adjusted using the ω parameters to account for positive samples. We use the open source code from https://github.com/ziyuanzhao2000/TNC_TS_baseline in our experiments.

TS2Vec: The method firstly creates two views, *i.e.*, a left one and a right one, on the overlapping subseries before encoding. In the latent space, the loss is calculated hierarchically from the finest granularity to the instance level *w.r.t.* both temporal contrastive loss and instance-wise contrastive loss. This mechanism enables TS2Vec to learn both coarse-grained and fine-grained representation to solve different downsteam tasks. We use the open source code from https://github.com/yuezhihan/ts2vec in our experiments.

CPC: Different from other methods listed in this section, CPC is not exclusively designed for time series data. On the contrary, this method can handle different forms of sequential input data, such as voice, video and time series. It implements a forecasting based pretext task in the latent space by using the representation of the past to forecast the future latent vectors. We use the open source code from https://github.com/Spijkervet/contrastive-predictive-coding in our experiments.

CoST: In order to capture the trend and seasonality explicitly, CoST contrasts in both time domain and frequency domain to learn their respective representation. It uses a trend feature disentangler, which is composed of a group of causal convolutional block and an average pooling layer, to learn a good trend representation. And the seasonal representation is learned on Fourier transformed sequences *w.r.t.* both amplitude and phase by the seasonal feature disentangler. We use the open source code from https://github.com/salesforce/CoST in our experiments.

TF-C: The method argues contrasting only in time domain is not enough, since a great amount of information is easier to capture in the frequency domain. It implements data augmentation, positive and negative pair selection in both domains.

The contrast is made in time space, frequency space and a time-frequency space, and the model is trained based on the contrastive loss in these three spaces. We use the open source code from https://github.com/mims-harvard/TFC-pretraining in our experiments.

E.2. Supervised Methods

DeepAR: The method builds a globally auto-regressive RNN on a large number of related time series and then performs probabilistic sampling based on the assumed probabilistic distribution, which can handle widely-varying scales through rescaling and velocity-based sampling and produce calibrated probabilistic forecasts. We use the open source code from https://github.com/husnejahan/DeepAR-pytorch in our experiments.

TCN: To achieve the fact that the network produces an output of the same length as the input, the method uses a 1D fully-convolutional network architecture where each hidden layer is the same length as the input layer. To accomplish the fact that there can be no leakage from the future into the past, the method uses causal convolutions where an output at time t is convolved only with elements from time t and earlier in the previous layer. In our experiments, we use the open source code from https://github.com/rajatsen91/deepglo, where the TCN is used for regularization.

DLinear: DLinear questions the effectiveness of emerging favored transformer-based solutions for the long term time series forecasting problem, which first decompose a time series into trend and remainder, and then only use a simple one-layer linear model for forecasting. DLinear is better than existing transformer-based LTSF solutions in terms of efficiency and effectiveness. In our experiments, we use the open source code from https://github.com/thuml/Time-Series-Library/, where the reproduction of DLinear can be found.

PatchTST: PatchTST is a patch-based algorithm for time series forecasting, which reduces the time and space complexity of the original Transformer by applying a patching technique,PatchTST also has the capability of learning from longer look-back windows and the capability of representation learning. Besides, Channel-independence is first used in transformer-based TSF area. In our experiments, we use the open source code from https://github.com/thuml/Time-Series-Library/, where the reproduction of PatchTST can be found.

TimesNet: Based on the multi-periodicity of time series, the TimesNet with a modular architecture is proposed to capture the temporal patterns derived from different periods. For each period, to capture the corresponding intraperiod- and interperiod-variations, a TimesBlock is designed within the TimesNet, which can transform the 1D time series into 2D space and simultaneously model the two types of variations by a parameter-efficient inception block. In our experiments, we use the open source code from https://github.com/thuml/Time-Series-Library/, where the reproduction of TimesNet can be found.

Autoformer: Autoformer is a decomposition architecture for time series forecasting. It consists of an Auto-Correlation mechanism and an inner decomposition block. The Auto-Correlation mechanism discovers dependencies and aggregates information at the series level, while the inner decomposition block decomposes the time series into multiple components. In our experiments, we use the open source code from https://github.com/thuml/Time-Series-Library/, where the reproduction of Autoformer can be found.

Informer: Informer is proposed to enhance the prediction capacity in the LSTF problem using a transformer-like model. The ProbSparse self-attention mechanism efficiently replaces the canonical self-attention, reducing time and memory usage. The self-attention distilling operation privilege dominant attention scores, reducing space complexity. The generative style decoder enables long sequence output with cumulative error. In our experiments, we use the open source code from https://github.com/thuml/Time-Series-Library/, where the reproduction of Informer can be found.

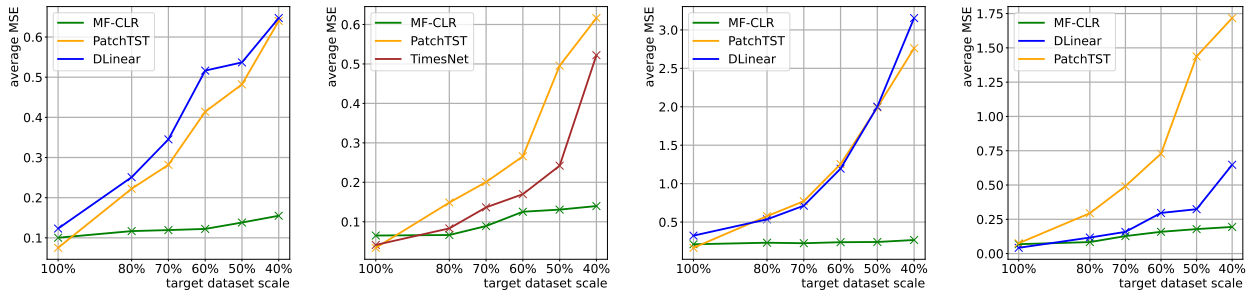
FEDformer: A frequency enhanced decomposed Transformer architecture with mixture of experts for seasonal-trend decomposition is proposed in order to better capture global properties of time series. By randomly selecting a fixed number of Fourier components, the proposed model achieves linear computational complexity and memory cost. In our experiments, we use the open source code from https://github.com/thuml/Time-Series-Library/, where the reproduction of FEDformer can be found.

F. Full Experimental Results

F.1. Forecasting

Table 9 summarizes the average performance across four datasets *w.r.t.* short-term forecasting and long-term forecasting using MSE as the metric in comparison with the supervised SOTAs.

To strengthen the results in Section 4.6.4, we also conduct transfer learning on both the short-term and the long-term forecasting tasks. The two transfer learning settings are the same with what in Section 4.6.4, *i.e.*, ETTm1 → ETTm2 and ETTm2 → ETTm1.For each case, two supervised methods with the lowest MSE are picked and the target dataset scale is decreased from 100% to 40%. We can conclude from Figure 7 that MF-CLR can deliver a much stable performance on both tasks and the it achieves the best performance after the target dataset scale is cut to 80% and below.



(a) STF, ETTm1 → ETTm2. (b) STF, ETTm2 → ETTm1. (c) LTF, ETTm1 → ETTm2. (d) LTF, ETTm2 → ETTm1.

Figure 7. Transfer learning for short-term forecasting the long-term forecasting. The STF and LTF refer to short-term forecasting and long-term forecasting, respectively.

Full results of the forecasting task compared with self-supervised baselines are summarized in Table 10. Note that the table contains results of both short-term forecasting and long-term forecasting.

Full results of the forecasting task compared with supervised baselines are summarized in Table 11. Note that the table contains results of both short-term forecasting and long-term forecasting.

F.2. Anomaly Detection

Full results of the anomaly detection task are summarized in Table 12.

F.3. Imputation

Full results of the imputation task are summarized in Table 13.

Table 8. Details about the preprocessing on 30 UEA datasets.

datasets	resample strategy
ArticularyWordRecognition	d_1-d_3: keep original; d_4-d_6: resample every 2 timestamps; d_7-d_9: resample every 3 timestamps.
AtrialFibrillation	d_1: keep original; d_2: resample every 2 timestamps.
BasicMotions	d_1-d_3: keep original; d_4-d_6: resample every 2 timestamps.
CharacterTrajectories	d_1-d_2: keep original; d_3: resample every 2 timestamps.
Cricket	d_1-d_3: keep original; d_4-d_6: resample every 3 timestamps.
DuckDuckGeese	d_1-d_{500}: keep original; d_{501}-d_{1000}: resample every 2 timestamps; d_{1001}-d_{1345}: resample every 3 timestamps.
EigenWorms	d_1-d_3: keep original; d_4-d_6: resample every 4 timestamps.
Epilepsy	d_1-d_2: keep original; d_3: resample every 2 timestamps.
EthanolConcentration	d_1-d_2: keep original; d_3: resample every 17 timestamps.
ERing	d_1-d_2: keep original; d_3-d_4: resample every 5 timestamps.
FaceDetection	d_1-d_{72}: keep original; d_{73}-d_{144}: resample every 2 timestamps.
FingerMovements	d_1-d_{19}: keep original d_{20}-d_{28}: resample every 2 timestamps
HandMovementDirection	d_1-d_5: keep original d_6-d_8: resample every 2 timestamps d_9-d_{10}: resample every 4 timestamps
Handwriting	d_1-d_2: keep original d_3: resample every 2 timestamps
Heartbeat	d_1-d_{21}: keep original d_{22}-d_{41}: resample every 3 timestamps d_{42}-d_{61}: resample every 5 timestamps
InsectWingbeat	d_1-d_{100}: keep original d_{101}-d_{200}: resample every 2 timestamps
JapaneseVowels	d_1-d_6: keep original d_7-d_{12}: resample every 2 timestamps
Libras	d_1: keep original d_2: resample every 3 timestamps
LSST	d_1-d_2: keep original d_3-d_4: resample every 2 timestamps d_5-d_6: resample every 3 timestamps
MotorImagery	d_1-d_{16}: keep original d_{17}-d_{32}: resample every 2 timestamps d_{33}-d_{48}: resample every 3 timestamps d_{49}-d_{64}: resample every 10 timestampss
NATOPS	d_1-d_{12}: keep original d_{13}-d_{24}: resample every 3 timestamps
PenDigits	d_1: keep original d_2: resample every 2 timestamps
PEMS-SF	d_1-d_{321}: keep original d_{322}-d_{642}: resample every 2 timestamps d_{643}-d_{963}: resample every 3 timestamps
Phoneme	d_1-d_6: keep original d_7-d_{11}: resample every 7 timestamps
RacketSports	d_1-d_3: keep original d_4-d_6: resample every 2 timestamps
SelfRegulationSCP1	d_1-d_2: keep original d_3-d_6: resample every 4 timestamps
SelfRegulationSCP2	d_1-d_2: keep original d_3-d_7: resample every 4 timestamps
SpokenArabicDigits	d_1-d_6: keep original d_7-d_{13}: resample every 3 timestamps
StandWalkJump	d_1-d_2: keep original d_3-d_4: resample every 5 timestamps
UWaveGestureLibrary	d_1: keep original d_2: resample every 3 timestamps d_3: resample every 5 timestamps

Table 9. Performance for short-term and long-term forecasting compared with supervised methods.

Methods	traffic	ETTm1	ETTm2	weather
		short-term forecasting		
MF-CLR	0.7033	0.0479	0.1227	0.1635
DLinear	1.0600	0.0435	1.1231	0.5421
PatchTST	1.0227	**0.0340**	**0.0752**	0.7513
TimesNet	1.1451	0.0413	0.1761	0.5686
Autoformer	**0.6410**	0.0429	0.2932	0.2380
Informer	1.5604	0.3700	0.2891	**0.1010**
FEDformer	1.4966	0.1028	0.1325	0.4733
		long-term forecasting		
MF-CLR	**0.7466**	0.0495	0.2060	0.5496
DLinear	1.0839	**0.0422**	0.3235	0.4866
PatchTST	0.7687	0.0754	**0.1653**	0.4523
TimesNet	1.0426	0.0835	0.4500	**0.4167**
Autoformer	1.0107	0.1239	0.4170	2.3023
Informer	3.2487	0.2685	0.7075	1.0272
FEDformer	0.8020	0.1005	0.5235	2.6785

Table 10. Full results of the forecasting task compared with self-supervised baselines.

			MF-CLR		T-Loss		TS-TCC		TNC		CPC		TS2Vec		CoST		TF-C	
metrics			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
traffic	h	168	0.1932	0.3181	0.1419	0.2698	0.2468	0.3619	1.5407	1.0837	0.4418	0.5556	0.2373	0.3825	0.1701	0.3161	0.9642	0.8212
		336	0.2711	0.3280	0.4138	0.4435	0.3954	0.4306	2.1240	1.2441	1.7449	1.1176	0.6018	0.5495	0.3573	0.4070	1.7345	1.1081
		ave.	**0.2322**	**0.3231**	0.2779	0.3567	0.3211	0.3963	1.8324	1.1639	1.0934	0.8366	0.4196	0.4660	0.2637	0.3616	1.3494	0.9647
	d	7	1.2134	0.8500	1.2938	0.9198	0.9944	0.9100	1.8577	1.0749	1.5475	0.8956	1.5226	0.9992	1.7506	1.0821	0.8794	0.8302
		14	1.2221	0.8950	1.4953	0.9937	1.9519	1.2322	2.0024	1.1857	1.6280	1.0268	1.0380	0.8411	1.6199	1.0655	2.6145	1.4415
		ave.	**1.2178**	**0.8725**	1.3946	0.9568	1.4732	1.0711	1.9301	1.1303	1.5878	0.9612	1.2803	0.9202	1.6853	1.0738	1.7470	1.1359
ETTm1	q	48	0.0402	0.1940	0.0135	0.1174	0.0160	0.1198	0.0493	0.1807	0.0385	0.1797	0.0078	0.0707	0.0061	0.0697	0.0634	0.1880
		672	0.0837	0.2240	0.0922	0.2433	0.0894	0.2430	0.2310	0.4594	0.1038	0.2427	0.1374	0.2697	0.1894	0.3284	0.4238	0.5996
		ave.	0.0620	0.2090	0.0529	**0.1804**	**0.0527**	0.1814	0.1402	0.3201	0.0712	0.2112	0.0726	0.1702	0.0978	0.1991	0.2436	0.3938
	h	24	0.0146	0.1111	0.0161	0.0759	0.0463	0.1823	0.0103	0.0748	0.0230	0.1210	0.0131	0.0991	0.0059	0.0590	0.2174	0.4015
		336	0.0602	0.1767	0.0701	0.2189	0.0645	0.1942	0.1139	0.2765	0.1104	0.2587	0.0748	0.2071	0.0717	0.2317	0.2583	0.4442
		ave.	**0.0374**	**0.1439**	0.0431	0.1474	0.0554	0.1883	0.0621	0.1757	0.0667	0.1899	0.0440	0.1531	0.0388	0.1454	0.2379	0.4229
	d	7	0.0530	0.1774	0.0731	0.2056	0.0779	0.2319	0.0979	0.2540	0.0281	0.1401	0.0821	0.2394	0.0782	0.2045	0.0611	0.2059
		30	0.0388	0.1631	0.0723	0.2422	0.0362	0.1616	0.0736	0.2176	0.1014	0.2514	0.1397	0.3440	0.0579	0.2342	0.7701	0.7076
		ave.	**0.0459**	**0.1703**	0.0727	0.2239	0.0570	0.1968	0.0858	0.2358	0.0647	0.1958	0.1109	0.2917	0.0681	0.2194	0.4156	0.4568
ETTm2	q	48	0.0480	0.2136	0.0274	0.1686	0.0277	0.1598	0.0507	0.1944	0.3273	0.5626	0.0453	0.1884	0.0335	0.1735	0.0684	0.1839
		672	0.2610	0.3967	0.2898	0.4929	0.2817	0.4655	0.3376	0.4915	0.8082	0.7526	0.2951	0.4595	0.3192	0.4719	0.2859	0.4317
		ave.	**0.1545**	**0.3052**	0.1586	0.3308	0.1547	0.3127	0.1942	0.3430	0.5678	0.6576	0.1702	0.3240	0.1764	0.3227	0.1772	0.3078
	h	24	0.0477	0.1843	0.0490	0.1747	0.0869	0.2464	0.0911	0.1523	0.1934	0.4080	0.0551	0.1907	0.0743	0.1495	0.1294	0.2726
		336	0.1715	0.3191	0.2144	0.3397	0.2059	0.3626	0.1947	0.3954	0.5982	0.6454	0.3359	0.4890	0.1758	0.3158	1.3143	1.0300
		ave.	**0.1096**	0.2517	0.1317	0.2572	0.1464	0.3045	0.1429	0.2739	0.3958	0.5267	0.1955	0.3399	0.1251	**0.2327**	0.7219	0.6513
	d	7	0.1340	0.2877	0.1578	0.2921	0.1360	0.2883	0.1866	0.3438	0.1323	0.2716	0.1102	0.2750	0.1472	0.2989	0.1197	0.2998
		30	0.2405	0.4209	0.2181	0.3903	0.6455	0.7516	0.9561	0.9289	0.3847	0.5654	0.3835	0.5554	0.2480	0.4413	0.3356	0.5102
		ave.	**0.1873**	0.3543	0.1880	**0.3412**	0.3908	0.5200	0.5714	0.6364	0.2585	0.4185	0.2469	0.4152	0.1976	0.3701	0.2277	0.4050
weather	m	144	0.0689	0.1405	0.7762	0.4533	0.0030	0.0337	0.0873	0.1555	0.0002	0.0100	0.0008	0.0249	0.2777	0.2662	0.0151	0.0710
		432	0.0779	0.0941	0.4273	0.2008	0.0024	0.0278	0.0896	0.1634	0.0004	0.0171	0.0006	0.0200	0.7536	0.2655	0.1525	0.1216
		ave.	0.0734	0.1173	0.6018	0.3271	0.0027	0.0308	0.0885	0.1595	**0.0003**	**0.0136**	0.0007	0.0225	0.5157	0.2659	0.0838	0.0963
	h	168	0.0198	0.0625	0.0341	0.0747	0.0120	0.0494	1.6593	0.4411	0.0015	0.0307	0.5264	0.3074	0.0529	0.0872	0.4379	0.2618
		720	0.2613	0.1102	0.8786	0.1770	0.0270	0.0661	0.0739	0.0878	0.0024	0.0379	0.5854	0.1381	0.6133	0.1696	0.0512	0.0710
		ave.	0.1406	0.0864	0.4564	0.1259	0.0195	0.0578	0.8666	0.2645	**0.0020**	**0.0343**	0.5559	0.2228	0.3331	0.1284	0.2446	0.1664
	d	7	0.4875	0.5602	0.3998	0.5771	0.5081	0.7011	0.6205	0.7672	0.7475	0.7618	0.3405	0.5069	0.3489	0.5601	0.5682	0.7404
		30	1.6040	0.8114	1.7850	0.8500	1.5844	0.8458	1.8956	0.9635	1.6989	0.9596	1.9746	0.9571	1.9500	0.9298	1.8244	0.9629
		ave.	**1.0458**	**0.6858**	1.0924	0.7136	1.0463	0.7735	1.2581	0.8654	1.2232	0.8607	1.1576	0.7320	1.1495	0.7450	1.1963	0.8517
	w	4	0.0071	0.0744	0.0107	0.0839	0.1084	0.2994	0.1019	0.2892	0.1215	0.3187	0.0150	0.1155	0.0187	0.1144	0.1251	0.3155
		13	0.3260	0.5061	0.1851	0.3217	0.7609	0.7816	1.9951	1.2794	0.1016	0.2531	0.4874	0.5903	0.6097	0.5651	2.0257	1.2959
		ave.	0.1666	0.2903	**0.0979**	**0.2028**	0.4347	0.5405	1.0485	0.7843	0.1116	0.2859	0.2512	0.3529	0.3142	0.3398	1.0754	0.8057

Table 11. Full results of the forecasting task compared with supervised baselines.

			MF-CLR		DeepAR		TCN		DLinear		PatchTST		TimesNet		Autoformer		Informer		FEDdformer	
metrics			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
traffic	h	168	0.1932	0.3181	0.1997	0.3390	1.3671	1.0029	0.1850	0.3490	0.3551	0.4765	0.5622	0.5973	0.3941	0.5528	1.7281	1.1002	0.4293	0.5762
		336	0.2711	0.3280	0.4411	0.5004	1.9195	1.1746	0.3160	0.3985	0.2689	0.3374	0.2817	0.3294	0.4334	0.4625	2.7394	1.3238	0.4983	0.5277
		ave.	**0.2322**	**0.3231**	0.3204	0.4197	1.6433	1.0887	0.2505	0.3737	0.3120	0.4069	0.4220	0.4633	0.4138	0.5077	2.2337	1.2120	0.4638	0.5519
	d	7	1.2134	0.8500	2.1155	1.0901	1.0333	0.7930	1.9349	0.9666	1.6902	0.9830	1.7279	0.9804	0.8879	0.6407	1.3927	1.1436	2.5639	1.4042
		14	1.2221	0.8950	1.5371	0.9898	1.3637	0.9652	1.8518	1.0948	1.2684	0.9047	1.8035	1.0621	1.5879	1.0408	3.7579	1.6398	1.1056	0.9204
		ave.	1.2178	0.8725	1.8263	1.0400	**1.1985**	0.8791	1.8934	1.0307	1.4793	0.9439	1.7657	1.0213	1.2379	**0.8408**	2.5753	1.3917	1.8348	1.1623
ETTm1	q	48	0.0402	0.1940	0.0057	0.0703	0.3931	0.6127	0.0065	0.0708	0.0101	0.0951	0.0084	0.0743	0.0056	0.0540	0.0148	0.1047	0.0404	0.1962
		672	0.0837	0.2240	0.5883	0.7196	0.6595	0.7877	0.0434	0.1420	0.0824	0.2336	0.1101	0.2623	0.0462	0.1689	0.2116	0.3640	0.0803	0.2257
		ave.	0.0620	0.2090	0.2970	0.3950	0.5263	0.7002	**0.0250**	**0.1064**	0.0462	0.1644	0.0593	0.1683	0.0259	0.1115	0.1132	0.2344	0.0604	0.2109
	h	24	0.0146	0.1111	0.0147	0.0988	0.7024	0.8292	0.0151	0.1064	0.0137	0.1038	0.0037	0.0513	0.0371	0.1719	0.0221	0.1274	0.0638	0.2177
		336	0.0602	0.1767	0.1733	0.3523	0.4443	0.6048	0.0409	0.1385	0.0687	0.1861	0.0955	0.2502	0.1962	0.3582	0.1593	0.3498	0.0698	0.1995
		ave.	0.0374	0.1439	0.0940	0.2256	0.5733	0.7170	**0.0280**	**0.1225**	0.0412	0.1450	0.0496	0.1507	0.1166	0.2650	0.0907	0.2386	0.0668	0.2086
	d	7	0.0530	0.1774	0.2745	0.4731	0.3626	0.5481	0.1091	0.3058	0.0297	0.1337	0.0429	0.1527	0.0825	0.2403	1.2313	1.0706	0.2266	0.4184
		30	0.0388	0.1631	0.7248	0.7666	0.0532	0.1730	0.0434	0.1635	0.0821	0.2379	0.0715	0.2049	0.0515	0.1908	0.3776	0.5294	0.1312	0.2976
		ave.	**0.0459**	**0.1703**	0.4996	0.6198	0.2079	0.3605	0.0763	0.2346	0.0559	0.1858	0.0572	0.1788	0.0670	0.2156	0.8045	0.8000	0.1789	0.3580
ETTm2	q	48	0.0480	0.2136	0.0552	0.1564	0.0421	0.1755	0.0138	0.0875	0.0203	0.1082	0.0160	0.1085	0.0513	0.2122	0.1161	0.3289	0.0188	0.1255
		672	0.2610	0.3967	0.3868	0.4899	0.2565	0.4101	0.1882	0.3341	0.1001	0.2426	0.3990	0.4994	0.1972	0.3360	0.6187	0.7086	0.2092	0.3705
		ave.	0.1545	0.3052	0.2210	0.3231	0.1493	0.2928	0.1010	0.2108	**0.0602**	**0.1754**	0.2075	0.3040	0.1242	0.2741	0.3674	0.5187	0.1140	0.2480
	h	24	0.0477	0.1843	0.0084	0.0573	0.0720	0.2222	0.1324	0.2600	0.1204	0.2447	0.1531	0.2770	0.2097	0.3309	0.1910	0.3544	0.1622	0.3043
		336	0.1715	0.3191	0.2444	0.4226	0.7422	0.7165	0.1850	0.3471	0.2280	0.3790	0.3381	0.4949	0.3706	0.4974	1.1425	0.9668	0.2664	0.4342
		ave.	**0.1096**	0.2517	0.1264	**0.2399**	0.4071	0.4693	0.1587	0.3036	0.1742	0.3119	0.2456	0.3860	0.2902	0.4142	0.6668	0.6606	0.2143	0.3693
	d	7	0.1340	0.2877	0.1109	0.2743	0.1483	0.3312	0.1579	0.2898	0.0598	0.1932	0.1363	0.2925	0.7144	0.7353	0.2307	0.3968	0.1399	0.2905
		30	0.2405	0.4209	0.2913	0.4788	0.2419	0.3917	0.4619	0.6041	0.1025	0.2600	0.5618	0.6908	0.4634	0.5676	0.2725	0.4272	0.7805	0.8318
		ave.	0.1873	0.3543	0.2011	0.3766	0.1951	0.3615	0.3099	0.4470	**0.0812**	**0.2266**	0.3491	0.4917	0.5889	0.6515	0.2516	0.4120	0.4602	0.5612
weather	m	144	0.0689	0.1405	0.0104	0.0375	0.3275	0.5325	0.0021	0.0451	0.0002	0.0079	0.0001	0.0077	0.3557	0.4485	0.0096	0.0862	0.0079	0.0757
		432	0.0779	0.0941	0.0526	0.2008	0.2813	0.2114	0.0024	0.0479	0.0002	0.0109	0.0003	0.0113	0.0265	0.1283	0.0012	0.0305	0.0036	0.0492
		ave.	0.0734	0.1173	0.0315	0.1192	0.3044	0.3720	0.0023	0.0465	0.0002	**0.0094**	**0.0002**	0.0095	0.1911	0.2884	0.0054	0.0584	0.0058	0.0625
	h	168	0.0198	0.0625	0.0053	0.0632	0.5266	0.6766	0.0032	0.0505	0.0025	0.0479	0.0024	0.0473	0.1313	0.2658	0.0733	0.1946	0.0238	0.1327
		720	0.2613	0.1102	0.1046	0.2949	0.6991	0.8098	0.0119	0.1026	0.0048	0.0557	0.0022	0.0342	0.0569	0.1898	0.0184	0.1118	0.0109	0.0847
		ave.	0.1406	0.0864	0.0550	0.1791	0.6128	0.74322	0.0075	0.0766	0.0037	0.0518	**0.0023**	**0.0408**	0.0941	0.2278	0.0458	0.1532	0.0173	0.1087
	d	7	0.4875	0.5602	0.5353	0.7140	0.4889	0.6876	2.1605	1.3557	3.0023	1.6231	2.2715	1.4023	0.4385	0.6330	0.3198	0.4270	1.8579	1.2431
		30	1.6040	0.8114	1.7550	0.9405	1.7375	0.9118	1.6071	0.8738	1.5758	0.9744	1.4263	0.8551	2.0147	1.0465	2.3069	1.0952	1.5587	0.8874
		ave.	**1.0458**	**0.6858**	1.1451	0.8273	1.1132	0.7997	1.8838	1.1147	2.2890	1.2987	1.8489	1.1287	1.2266	0.8397	1.3133	0.7611	1.7083	1.0653
	w	4	0.0071	0.0744	0.3902	0.6022	0.1663	0.3788	0.1708	0.3859	0.0343	0.1536	0.1847	0.3945	3.1155	1.9798	1.4644	0.9952	5.9462	2.3537
		13	0.3260	0.5061	0.1794	0.3859	0.1116	0.2787	0.1565	0.3241	0.1942	0.3146	0.0537	0.2160	4.0219	2.0210	0.3192	0.4976	3.1982	1.5834
		ave.	0.1666	0.2903	0.2848	0.4940	0.1390	0.3288	0.1637	0.3550	**0.1143**	**0.2341**	0.1192	0.3053	3.5687	2.0004	0.8918	0.7464	4.5722	1.9686

Table 12. Full results of the anomaly detection task.

	SMAP				MSL				SMD				SWaT			
metrics	Acc.	P	R	F1	Acc.	P	R	F1	Acc.	P	R	F1	Acc.	P	R	F1
MF-CLR	0.9805	0.6018	0.7068	0.6403	0.9428	0.4769	0.7685	0.5721	0.9689	0.4772	0.8003	0.5751	0.9833	0.9251	0.9387	0.9318
self-supervised																
T-Loss	0.8780	0.1201	0.1218	0.1209	0.9669	0.4851	0.5112	0.4854	0.9732	0.4960	0.6508	0.5443	0.9809	0.8677	0.9946	0.9268
TS-TCC	0.9786	0.5692	0.6069	0.5800	0.9497	0.5109	0.7541	0.5928	0.9404	0.3579	0.8624	0.4734	0.9803	0.9754	0.8593	0.9137
TNC	0.9786	0.5726	0.6618	0.6065	0.9501	0.4935	0.7096	0.5759	0.9400	0.3533	0.8548	0.4667	0.9817	0.9075	0.9459	0.9263
CPC	0.8487	0.1357	0.1647	0.1444	0.9386	0.2651	0.3461	0.2926	0.9288	0.3178	0.8165	0.4251	0.9876	0.8073	0.9470	0.8716
TS2vec	0.9818	0.6273	0.7780	0.6819	0.9393	0.4620	0.7594	0.5592	0.9282	0.3139	0.8920	0.4408	0.9872	0.9275	0.9703	0.9484
CoST	0.9788	0.5649	0.6676	0.6028	0.9354	0.4516	0.6696	0.5274	0.9307	0.3204	0.8916	0.4454	0.9879	0.9279	0.9763	0.9515
TF-C	0.9789	0.5854	0.6901	0.6235	0.9449	0.4813	0.7537	0.5705	0.9378	0.3554	0.8741	0.4688	0.9831	0.9741	0.8844	0.9271
parametric																
SPOT	0.7340	0.2263	0.5575	0.2572	0.8309	0.0936	0.1985	0.1021	0.7128	0.1059	0.5475	0.1565	0.9395	0.7648	0.7258	0.7448
DSPOT	0.7454	0.2736	0.6614	0.3218	0.7948	0.1318	0.2242	0.1354	0.4343	0.0531	0.6801	0.0919	0.8355	0.4034	0.7357	0.5211
Luminol	0.8743	0.2581	0.4383	0.2934	0.8174	0.1808	0.4707	0.2445	0.6669	0.0992	0.7024	0.1617	0.8932	0.5394	0.8356	0.6500
supervised																
DLinear	0.9312	0.3701	0.3418	0.3500	0.9408	0.2888	0.2123	0.2381	0.9846	0.7434	0.7223	0.7113	0.9816	0.9816	0.9816	0.9816
PatchTST	0.9407	0.4449	0.4089	0.4194	0.9378	0.2710	0.2113	0.2290	0.9856	0.7311	0.7248	0.7040	0.9809	0.9222	0.9210	0.9216
TimesNet	0.9292	0.3655	0.3400	0.3484	0.9362	0.2536	0.1917	0.2111	0.9835	0.7227	0.6973	0.6831	0.9807	0.9175	0.9246	0.9211
Autoformer	0.9287	0.3668	0.3246	0.3339	0.9362	0.2125	0.1673	0.1836	0.9858	0.7562	0.7205	0.7126	0.9582	1.0000	0.6562	0.7924
Informer	0.9239	0.2921	0.2634	0.2697	0.9366	0.2546	0.1966	0.2191	0.9858	0.7568	0.7205	0.7128	0.9581	1.0000	0.6555	0.7919
FEDformer	0.9239	0.3509	0.3139	0.3198	0.9362	0.2100	0.1671	0.1824	0.9859	0.7568	0.7237	0.7156	0.9581	1.0000	0.6555	0.7919

Table 13. Full results of the imputation task.

dataset	mask ratio	metric	MF-CLR	T-loss	TS-TCC	TNC	CPC	TS2Vec	CoST	TF-C	DLinear	PatchTST	TimesNet	Autoformer	Informer	FEDformer
traffic	12.5%	MSE	0.4366	0.4189	0.3733	0.3921	0.6650	0.3307	0.3413	0.4271	0.1486	0.9904	12.1985	0.5206	0.5383	0.5384
		MAE	0.4146	0.4180	0.3939	0.4121	0.5336	0.3735	0.3753	0.4326	0.2118	0.7172	1.4739	0.4425	0.4329	0.4552
	25.0%	MSE	0.4712	0.4243	0.4503	0.4106	0.6628	0.3798	0.3768	0.4824	0.2324	0.9357	2.2654	0.5422	0.5599	0.5300
		MAE	0.4364	0.4297	0.4339	0.4243	0.5438	0.3900	0.3974	0.4630	0.2705	0.6890	0.7923	0.4500	0.4366	0.4405
	37.5%	MSE	0.4884	0.4805	0.4875	0.4882	0.7488	0.4121	0.4281	0.5124	0.2661	0.9367	1.1240	0.5563	0.5961	0.5296
		MAE	0.4562	0.4628	0.4611	0.4526	0.5916	0.4172	0.4250	0.4755	0.3094	0.6898	0.6874	0.4602	0.4500	0.4369
	50.0%	MSE	0.5412	0.5289	0.5287	0.4877	0.7745	0.4874	0.4825	0.5745	0.3303	0.9356	0.7744	0.5853	0.6232	0.5530
		MAE	0.4878	0.4897	0.4850	0.4623	0.6056	0.4546	0.4571	0.5184	0.3525	0.6905	0.5622	0.4715	0.4526	0.4467
ETTm1	12.5%	MSE	0.0629	0.1120	0.1102	0.1052	0.1001	0.0760	0.0883	0.1008	0.0492	0.7300	0.5694	0.0529	0.0344	0.0322
		MAE	0.1548	0.2243	0.2233	0.2164	0.2275	0.1887	0.2010	0.2144	0.1428	0.6066	0.4885	0.1602	0.1094	0.1146
	25.0%	MSE	0.0750	0.0985	0.1138	0.1049	0.1118	0.0746	0.0860	0.1045	0.0627	0.6493	0.3794	0.1165	0.0364	0.0431
		MAE	0.1717	0.2098	0.2218	0.2158	0.2369	0.1875	0.1934	0.2130	0.1667	0.5165	0.4194	0.2528	0.1137	0.1338
	37.5%	MSE	0.0895	0.1252	0.1311	0.1240	0.1481	0.0912	0.0999	0.1297	0.0859	0.6460	0.3699	0.1615	0.0412	0.0550
		MAE	0.1967	0.2375	0.2457	0.2378	0.2703	0.2086	0.2121	0.2420	0.1944	0.4986	0.4074	0.3000	0.1231	0.1584
	50.0%	MSE	0.1132	0.1574	0.1846	0.1644	0.1948	0.1037	0.1132	0.1695	0.1128	0.6373	0.4023	0.1863	0.0476	0.0870
		MAE	0.2194	0.2713	0.2886	0.2756	0.3146	0.2251	0.2326	0.2800	0.2222	0.4910	0.4183	0.3121	0.1347	0.2104
ETTm2	12.5%	MSE	0.0289	0.0386	0.0439	0.0502	0.0411	0.0360	0.0349	0.0472	0.0241	0.1759	0.2477	0.0502	0.0150	0.0180
		MAE	0.1127	0.1404	0.1454	0.1524	0.1435	0.1353	0.1313	0.1472	0.1060	0.3060	0.3117	0.1512	0.0822	0.0929
	25.0%	MSE	0.0341	0.0442	0.0568	0.0547	0.0495	0.0405	0.0390	0.0529	0.0319	0.0931	0.0877	0.0350	0.0168	0.0222
		MAE	0.1270	0.1538	0.1650	0.1631	0.1621	0.1469	0.1387	0.1580	0.1246	0.2185	0.2062	0.1298	0.0861	0.1030
	37.5%	MSE	0.0429	0.0517	0.0736	0.0715	0.0720	0.0539	0.0479	0.0645	0.0408	0.0811	0.0763	0.0350	0.0190	0.0312
		MAE	0.1433	0.1689	0.1899	0.1865	0.1951	0.1717	0.1542	0.1779	0.1418	0.1984	0.1926	0.1313	0.0915	0.1253
	50.0%	MSE	0.0542	0.0596	0.1005	0.0944	0.1017	0.0628	0.0477	0.0938	0.0486	0.0796	0.0937	0.0663	0.0239	0.0545
		MAE	0.1602	0.1779	0.2196	0.2121	0.2318	0.1903	0.1603	0.2126	0.1553	0.1942	0.2150	0.1812	0.1058	0.1684
weather	12.5%	MSE	0.0071	0.0216	0.0282	0.0267	0.0502	0.0253	0.0170	0.0319	0.0123	0.1973	0.4185	0.0879	0.0128	0.0085
		MAE	0.0545	0.0989	0.1103	0.1091	0.1690	0.1061	0.0875	0.1180	0.0700	0.3230	0.4272	0.2096	0.0764	0.0595
	25.0%	MSE	0.0121	0.0265	0.0399	0.0463	0.0859	0.0344	0.0247	0.0430	0.0203	0.0839	0.1295	0.0183	0.0191	0.0143
		MAE	0.0726	0.1103	0.1392	0.1434	0.2152	0.1303	0.1049	0.1377	0.0943	0.1931	0.2398	0.0895	0.0930	0.0752
	37.5%	MSE	0.0156	0.0354	0.0625	0.0595	0.1150	0.0388	0.0338	0.0609	0.0279	0.0545	0.1172	0.0228	0.0208	0.0214
		MAE	0.0902	0.1249	0.1734	0.1675	0.2563	0.1365	0.1240	0.1675	0.1101	0.1336	0.2253	0.0932	0.0967	0.0936
	50.0%	MSE	0.0286	0.0458	0.0918	0.0968	0.1339	0.0467	0.0341	0.0985	0.0355	0.0516	0.0879	0.0360	0.0205	0.0430
		MAE	0.1180	0.1431	0.2093	0.2179	0.2767	0.1570	0.1302	0.2170	0.1267	0.1273	0.2018	0.1232	0.0951	0.1434