SimTS: Rethinking Contrastive Representation Learning for Time Series Forecasting

Xiaochen Zheng [*1 2] **Xingyu Chen** [*3] **Manuel Schürch** [1 2] **Amina Mollaysa** [1 2] **Ahmed Allam** [1]
Michael Krauthammer [1 2]

Abstract

Contrastive learning methods have shown an impressive ability to learn meaningful representations for image or time series classification. However, these methods are less effective for time series forecasting, as optimization of instance discrimination is not directly applicable to predicting the future state from the history context. Moreover, the construction of positive and negative pairs in current technologies strongly relies on specific time series characteristics, restricting their generalization across diverse types of time series data. To address these limitations, we propose SimTS, a simple representation learning approach for improving time series forecasting by learning to predict the future from the past in the latent space. SimTS does not rely on negative pairs or specific assumptions about the characteristics of the particular time series. Our extensive experiments on several benchmark time series forecasting datasets show that SimTS achieves competitive performance compared to existing contrastive learning methods. Furthermore, we show the shortcomings of the current contrastive learning framework used for time series forecasting through a detailed ablation study. Overall, our work suggests that SimTS is a promising alternative to other contrastive learning approaches for time series forecasting.

1. Introduction

The field of time series forecasting has experienced significant progress in recent years with a wide range of practical applications across different sectors such as finance (Sezer et al., 2020), traffic (Zheng & Huang, 2020), and clinical practice (Johnson et al., 2016). The availability of large volumes of data is one of the key factors behind these advancements. In particular, self-supervised learning approaches such as contrastive learning (Yue et al., 2022; Woo et al., 2022; Yèche et al., 2021) have shown promise in exploiting these datasets and have continually outperformed supervised approaches (Bai et al., 2018; Salinas et al., 2020; Zhou et al., 2021) in time series forecasting tasks. Self-supervised contrastive approaches learn representations by mapping similar instances (i.e., positive pairs) to similar representations while pushing dissimilar instances (i.e., negative pairs) apart. Most contrastive learning approaches rely on instance discrimination (Wu et al., 2018). The resulting representations contain information that can discriminate well between different instances of time series, making them informative for downstream tasks such as time series *classification*. However, in time series *forecasting*, the goal is to predict the future based on past time windows rather than discriminating between instances. Consequently, features learned by instance discrimination may not be sufficient for accurate forecasting.

Additionally, identifying positive and negative pairs for time series forecasting is challenging. Contrastive learning relies on data augmentations to generate positive pairs. While it is possible to find semantic preserving augmentations for time series classification (Ye & Keogh, 2009; Yèche et al., 2021; Nonnenmacher et al., 2022), it is more difficult to identify augmentation methods that can be generalized to time series forecasting. Besides, most existing methods for constructing negative pairs depend heavily on the individual characteristics of the time series, making them not applicable to other types of time series. Yue et al. (2022); Kiyasseh et al. (2021); Yèche et al. (2021); Hyvarinen & Morioka (2016); Tonekaboni et al. (2021) propose methods based on the assumptions that (1) the similarity between segments of the same time series decreases as the time lag increases, and (2) segments of distinctive time series are dissimilar. However, particular time series do not adhere to these assumptions, resulting in unsatisfactory representations (Eldele et al., 2021; Nonnenmacher et al., 2022) in other time series. For instance, in a time series with a strong periodicity, similar patterns exist between or within instances. As illustrated

[*]Equal contribution [1]University of Zurich, Zurich, Switzerland [2]ETH AI Center, Zurich, Switzerland [3]ETH Zurich, Zurich, Switzerland. Correspondence to: Xiaochen Zheng <xiaochen.zheng@uzh.ch>.

Preprint. Work in progress.

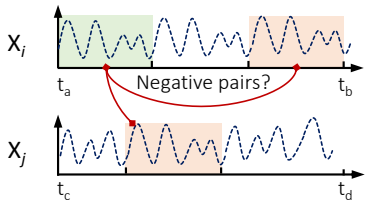


Figure 1. Problems with selecting negative pairs based on methods proposed in (Yèche et al., 2021; Yue et al., 2022; Woo et al., 2022) when cross-instance and cross-time repeated patterns exist.

in Figure 1, selecting times-windows randomly may result in selecting inappropriate negative pairs (Tian et al., 2020), leading to false repulsion, where the model incorrectly discriminates representations of similar samples. Other recent approaches are based on disentanglement (Woo et al., 2022; Wang et al., 2022) or fusion (Yang & Hong, 2022; Zhang et al., 2022b), assuming that a time series can be represented by trends and seasonality components. As a result, these approaches may not generalize well across various forecasting datasets, since real-world data often lack consistent seasonality. In general, this reliance on specific characteristics limits their generalizability when applied to different types of time series data, which will be demonstrated through detailed experiments in Section 4.

To address these limitations in contrastive representation learning for time series forecasting, the paper aims to answer the following key question: "what is important for time series forecasting with contrastive learning, and how can we adapt contrastive ideas more effectively to time series forecasting tasks?" Beyond contrastive learning, we propose a ***Sim**ple Representation Learning Framework for **T**ime **S**eries Forecasting* (**SimTS**), which is inspired by predictive coding (Oord et al., 2018): we learn a representation such that the latent representation of the future time windows can be predicted from the latent representation of the history time windows. In particular, we build upon a siamese network structure (Bromley et al., 1993; Chen & He, 2021) and propose key refinements that enable better prediction performance with a simpler model structure compared to state-of-art methods. First, we divide a given time series into *history* and *future* segments and then use an encoding network to map them to their latent space. Second, we use a predictive layer to predict the latent representation of the *future* segment from the *history* segment. We regard the predicted representation (from the *history* segment) and the encoded representation of the *future* segment as positive pairs. The representations learned in this way encode features that are useful for forecasting tasks.

Moreover, the paper questions existing assumptions and techniques used for constructing positive and negative pairs. We provide a detailed discussion and several experiments showing their shortcomings when applied to various time series. Specifically, inspired by (Chen & He, 2021; Grill et al., 2020; Tian et al., 2021), we question the proposed usage of negative pairs for time series forecasting and the idea of augmenting the data to generate positive pairs, which is empirically investigated in several experiments with different contrastive methods. As a consequence, our model does not use negative pairs to avoid false repulsion. We hypothesize that the most important mechanism behind representation learning for time series forecasting is maximizing the shared information between representations of *history* and *future* time windows. In our proposed model, we explicitly impose a constraint that the learned representation of history should encode as much information as possible by predicting the latent representation of the future from the latent representation of history. This mechanism simplifies several existing approaches and leads to state-of-the-art forecasting results, as thoroughly demonstrated in this paper.

Our contributions can be summarized as follows:

- We propose a novel method (SimTS) for time series forecasting, which employs a siamese structure and a simple convolutional encoder to learn representations in latent space without requiring negative pairs.
- We demonstrate the effectiveness and generalizability of SimTS across various types of time series through experiments on multiple types of benchmark datasets. Our method outperforms state-of-the-art methods for multivariate time series forecasting.
- We conduct extensive ablation experiments to assess and evaluate the effectiveness of various assumptions that are widely used in current state-of-the-art contrastive learning frameworks. This provides insights into the key factors that contribute to the performance of time series forecasting and sheds light on potential areas for improvement in future research.

2. Related Works

Researchers have recently developed numerous deep learning models to address the challenges of time series forecasting. Traditional models for time series prediction, such as ARIMA (Liu et al., 2016), SVM (Han et al., 2012), and VAR (Box et al., 2015), have been outperformed on many datasets by deep learning models, including RNN (Wen et al., 2017), CNN (Bai et al., 2018) and transformers (Vaswani et al., 2017). TCN (Bai et al., 2018) introduces dilated convolutions (Oord et al., 2016) for time series forecasting, which incorporates dilation factors into conventional CNNs to increase the receptive field significantly. To improve the effectiveness of long-term time series

forecasting, the conventional transformer is modified and applied to time series: LogTrans (Li et al., 2019) suggests the *LogSparse* attention; Informer (Zhou et al., 2021) develops the *ProbSparse* self-attention mechanism to reduce the computational cost of long-term forecasting.

Recent developments in self-supervised learning have successfully discovered meaningful representations for images (He et al., 2020; Chen et al., 2020) with InfoNCE loss (Oord et al., 2018). To get reliable time-series representations, several approaches have been investigated. Some studies focus on formulating time segments as contrastive pairs: ICA (Hyvarinen & Morioka, 2016) investigates non-stationarity in temporal data to find a representation that allows optimal time segment discrimination; TNC (Tonekaboni et al., 2021) establishes a temporal neighborhood to contrast between neighboring segments and learn the underlying temporal dependency of non-stationary time series. However, these methods do not perform as well in forecasting tasks since they focus on extracting neighborhood features and fail to capture global patterns in time series. Furthermore, some methods utilize more complicated contrastive learning approaches to learn effective representations for time series. For example, (Franceschi et al., 2019) learns scalable representations for various time series lengths using contrasting positive, negative, and reference pairs with an innovative triplet loss. TS2Vec (Yue et al., 2022) employs hierarchical contrastive learning over time series augmentations, generating representations for each time step. However, these approaches formulate contrastive learning frameworks as classification tasks, which try to learn representations by discriminating time series from different classes and therefore ignore learning predictive features. Additionally, as time series can be (re-)constructed by combining trend, season, and noise components (Shumway et al., 2000), there is growing research that uses time series decomposition in unsupervised learning. CoST (Woo et al., 2022) encodes disentangled trend and seasonal representations using contrastive learning. BTSF (Yang & Hong, 2022) aggregates time and spectral domain to extract global information and refine representations. While decomposition-related methods may exhibit robust performance in certain datasets, they heavily rely on underlying assumptions about the data's characteristics and tend to fail when dealing with datasets that lack specific seasonality or trend.

3. Methods

3.1. Motivation

In this work, we rethink "what is important for time series forecasting with contrastive learning?" Firstly, we observe that the existing methods might (1) ignore the possibility that repeated patterns exist within a time series, even though they may be located far apart from one another, and (2) disregard the possibility that distinct time series may contain similar patterns. We aim to identify a more suitable design that considers the inherent nature of time series forecasting and adheres to necessary assumptions for effective representation learning. We argue that a good representation should effectively capture the temporal dependencies between past segments and future predictions in forecasting tasks, emphasizing that the temporal differences hold greater significance than the similarity between positive and negative pairs. Thus, we design predictive positive pairs that can learn more flexible and adaptive representations.

Secondly, current approaches require sufficient negative pairs to avoid collapsing (Chen et al., 2020; Chen & He, 2021; Zhang et al., 2022a). Collapsing happens in Siamese networks (Bromley et al., 1993) where the model produces a constant representation regardless of the input. Although the introduction of negative pairs constrains the solution space and prevents collapsing, it might also induce the issue of false repulsion. Simultaneously, identifying suitable augmentation methods and negative pairs for forecasting tasks can be challenging, especially when repeated patterns exist across different samples. Such challenges motivate us to explore alternative approaches that circumvent negative pairs and implement stop-gradient solutions.

Furthermore, we contend that real-world data often lack distinct seasonality, making it difficult for models to learn irregular temporal information using abstract features. Our experiments demonstrate that learned representations, which discard some additional model components, yield better forecasting performance than the state-of-the-art contrastive model CoST (Woo et al., 2022), as shown in Table 5. These results suggest that current methods may not generalize well to diverse time series datasets. Finally, it leads us to the central motivation of our SimTS model: we train an encoder to learn time series representations by predicting its future from historical segments in the latent space. SimTS achieves the best performance in time series forecasting benchmark datasets with a relatively simpler design compared to other contrastive learning frameworks.

3.2. SimTS: Simple Representation Learning for Time Series Forecasting

Given a time series $X = [x_1, x_2, \ldots, x_T] \in \mathbb{R}^{C \times T}$, where C is the number of features (i.e., variables) and T denotes the sequence length. Our objective is to learn a latent representation of the *history* segment $X^h = [x_1, x_2, \ldots, x_K]$, where $0 < K < T$, such that our model can predict the *future* segment $X^f = [x_{K+1}, x_{K+2}, \ldots, x_T]$ from it.

Inspired by well-developed contrastive learning frameworks (Oord et al., 2018; Chen et al., 2020; Grill et al., 2020; Chen & He, 2021), SimTS learns time series representations by maximizing the similarity between predicted

and encoded latent features for each timestamp. The approach involves designing an encoder network, denoted as F_θ, which maps historical and future segments to their corresponding latent representations, Z^h and Z^f, respectively. The encoder's objective is to learn an informative latent representation $Z^h = F_\theta(X^h) = [z_1^h, z_2^h, ..., z_K^h] \in \mathbb{R}^{C' \times K}$ that can be used to predict the latent representation of the future through a prediction network. The SimTS model consists of four main parts:

- A siamese neural network architecture (Bromley et al., 1993; Chen & He, 2021) consisting of two identical networks that share parameters. The time series is divided into the *history* segment X^h, and *future* segment X^f, and given as inputs to the siamese network. The siamese network learns to map them to their latent representations Z^h, Z^f.
- A multi-scale encoder consisting of a projection layer that projects raw features into a high dimensional space and multiple CNN blocks with different kernel sizes.
- A predictor network G_ϕ that takes the last column of the encoded *history* view as input and predicts the *future* in latent space.
- A cosine similarity loss that only takes positive samples into account.

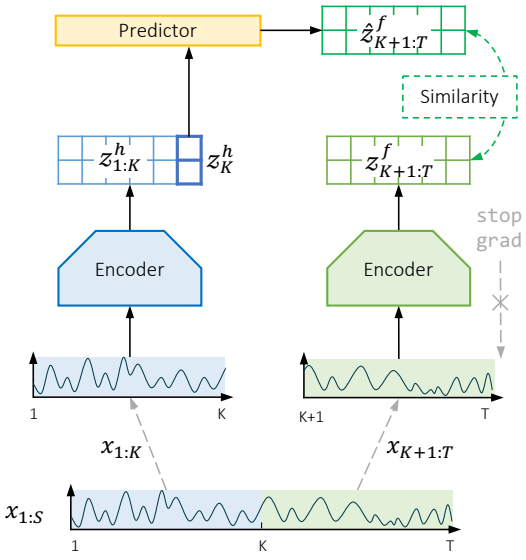


Figure 2. Illustration of our proposed SimTS.

Figure 2 depicts the overall architecture of SimTS. Our model architecture consists of two paths: the *history* encoding path and the *future* encoding path. The *history* encoding path takes the *history* view X^h and outputs $Z^h = F_\theta(X^h)$. The *future* encoding path takes the *future* view X^f and outputs the encoded latent representation of the future $Z^f = F_\theta(X^f) = [z^f_{K+1}, z^f_{K+2}, ..., z^f_T] \in \mathbb{R}^{C' \times (T-K)}$. As proposed in (Grill et al., 2020; Zeng et al., 2022), we apply a predictive MLP network G_ϕ on the last column of Z^h, denoted as z^h_K, to predict the *future* latent representations: $\hat{Z}^f = G_\phi(z^h_K) = [\hat{z}^f_{K+1}, \hat{z}^f_{K+2}, ..., \hat{z}^f_T] \in \mathbb{R}^{C' \times (T-K)}$. Intuitively, the last column allows the encoder to condense the history information into a summary by properly choosing the kernel size. The training objective is to attract the *predicted future* and *encoded future* timestamps in representation space without introducing the negative pairs. As the predicted future latent representation is learned from the latent representation of the history, by forcing the predicted latent representation of the future to be close to the encoded latent representation of the future, we are forcing the model to learn a representation of the history that is informative for the future. Therefore, we regard the encoded Z^f and the predicted *future* representations $\hat{Z}^f$ as the positive pair and calculate the negative cosine similarity between them:

$$Sim(\hat{Z}^f, Z^f) = -\frac{1}{T-K} \sum_{i=K+1}^{T} \frac{\hat{z}^f_i}{\| \hat{z}^f_i \|_2} \cdot \frac{z^f_i}{\| z^f_i \|_2}, \quad (1)$$

where $\| \cdot \|_2$ is l_2-norm and $Sim(\cdot)$ is the average cosine similarity of all time steps. Algorithm 1 summarises the proposed SimTS.

3.3. Multi-Scale Encoder

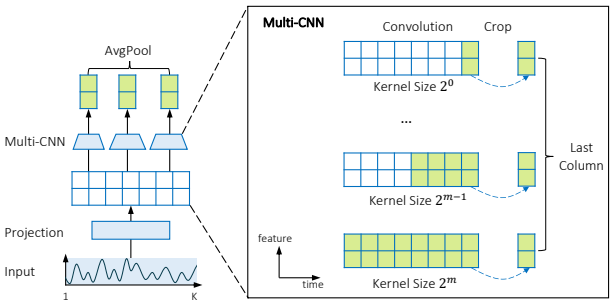


Figure 3. Multi-scale encoder. Composed of a projection layer and a set of parallel 1d convolutions with kernel size 2^i, for $i \in \{0, 1, ..., m\}$. An averaged pooling layer is added on the top of convolutions.

To learn a meaningful representation, the structure of the encoder network F_θ plays a vital role. Given the nature of time series, we would like our base encoder F_θ to extract temporal (inter-time) dependency from local and global patterns. For short-term forecasting, shorter local patterns (i.e., motifs) are ideal, whereas, for long-term forecasting, longer sets of global patterns are preferred. Therefore, we propose to use a convolutional network with multiple filters

Algorithm 1 SimTS's PyTorch-like Pseudocode

initialize θ, ϕ
given a mini-batch $\mathcal{D} = \{X_i\}_{i\in[1:N]}$ with N samples
for X **in** $\mathcal{D}$ **do**
  X^h, X^f = X [:, : K, :], X [:, K :, :]
  $Z^h, Z^f = F_\theta(X^h), F_\theta(X^f)$
  $\hat{Z}^f = G_\phi(Z^h[:, K, :])$
  $\hat{Z}^f$ = normalize($\hat{Z}^f$)
  Z^f = normalize(Z^f).detach()
  L = - $(\hat{Z}^f \cdot Z^f)$.mean()
  L.backward()
  update(F_θ, G_ϕ)
end for

that have various kernel sizes, which can extract both global and local patterns.

Figure 3 illustrates the details of the encoder F_θ. First, each time series input is passed through a convolutional projection layer. The projection layer enables us to project time series into a latent space (Yue et al., 2022; Woo et al., 2022; Wang et al., 2022). We aim to capture abstract information and consistent intra-time relationships between features that may not be immediately apparent from the raw data. So that the model can potentially learn more informative and abstract representations of the raw inputs. Second, for a time series X with length K, we have $m = [log_2 K] + 1$ parallel convolution layers on the top of the projection layer, and the ith convolution has kernel size 2^i, where $i \in \{0, 1, ..., m\}$. These different kernel sizes can extract corresponding local/global patterns. Each convolution i takes the latent features from the projection layer and generates a representation $\hat{Z}_{(i)}$. The final multi-scale representation Z are obtained by averaging across $\hat{Z}_{(0)}, \hat{Z}_{(1)}, ..., \hat{Z}_{(m)}$.

3.4. Stop-gradient Operation

We apply a stop-gradient operation (Chen & He, 2021) to the *future* encoding path in our model. Considering that we learn to encode both *history* and *future* using the same encoder, the model may optimize the encoder by pushing encoded *future* Z^f towards the predicted *future* $\hat{Z}^f$. As the encoder should constrain the latent of the past to be predictive of the latent of the *future*, only $\hat{Z}^f$ can only move towards Z^f in the latent space, not vice versa (Zhang et al., 2022a). Due to the stop-gradient operation on Z^f, our encoder cannot receive updates from *future* representations Z^f and is constrained to only optimize the *history* representation and its prediction $\hat{Z}^f$. With stop-gradient ($\mathtt{sg}$), the loss is:

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(X^h, X^f) &= Sim\left(G_\phi\left(F_\theta(X^h)\right), F_{\mathtt{sg}(\theta)}(X^f)\right) \\ &= Sim(\hat{Z}^f, \mathtt{sg}(Z^f))\end{aligned} \tag{2}$$

The loss in definition (2) is for one sample $X = [X^h, X^f]$. The loss for a mini-batch $\mathcal{D} = \{X_i^h, X_i^f\}_{i\in[1:N]}$ can be written as

$$\mathcal{L}_{\theta,\phi}(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{\theta,\phi}(X_i^h, X_i^f), \tag{3}$$

which corresponds to the average loss across all samples in the mini-batch.

4. Experiments

As our goal is to learn a meaningful representation for various types of time series data for forecasting tasks, we focus on experimental settings where we can test the representation power of our model on various forecasting benchmark datasets. To keep a fair comparison, we follow the exact same setup as in CoST and TS2Vec. We first use our trained model to obtain the latent representation of the time series, then train a ridge regression model on the learned latent representation for forecasting, i.e., predicting future L time steps.

4.1. Datasets and Baselines

We compare our method to the most recent state-of-the-art two-stage representation learning methods for time series as well as to end-to-end learning methods where the model includes both the representation learning part and forecasting part are trained in an end-to-end fashion. The representation learning approaches include TS2Vec (Yue et al., 2022), CoST (Woo et al., 2022) and TNC (Tonekaboni et al., 2021) and end-to-end models include Informer (Zhou et al., 2021) and LogTrans (Li et al., 2019), and two-stage representation learning approaches include TS2Vec, CoST and TNC. The details and implementations of the baselines are provided in the appendix. Our model was tested for both univariate and multivariate forecasting. In the case of a dataset with C features, we either predict the future values for all C features (i.e., multivariate forecasting) or only focus on forecasting the future values of one specific feature (univariate forecasting).

Our experiments are carried out on six real-world public benchmark datasets. **Electricity Transformer Temperature (ETT)** (Zhou et al., 2021) measures long-term deployment of electric power. It consists of two hourly-sampled datasets (ETTh) and two 15-minute-sampled datasets (ETTm), which are collected for 2 years and from two different Chinese provinces. ETT datasets contain one

oil temperature feature and six power load features. In univariate forecasting, we only take oil temperature to train and forecast. In multivariate forecasting, we employ all features in our training and prediction. **Exchange-Rate**[1] (Lai et al., 2018) contains the daily exchange rates of eight foreign countries from 1990 to 2016, including Australia, Britain, Canada, Switzerland, China, Japan, New Zealand, and Singapore. We consider the values of Singapore for univariate forecasting and all countries' value for multivariate forecasting. **Weather**[2] consists of local climatological data for almost 1,600 U.S. areas for 4 years. The data is collected every 10 minutes. Each time step contains 11 weather variables and one target feature, 'Wet Bulb Celsius.' In univariate forecasting, we only consider the feature 'Wet Bulb Celsius'; in multivariate forecasting, all features are included. The detailed statistics of the datasets are in the appendix (Table 6).

4.2. Experimental setup

We divide all datasets into training, validation, and test sets in the ratio of 6:2:2. Throughout the evaluation stage, the model parameters are frozen to output representations.

The input time series are projected to a 64-dimensional latent space using a convolutional projector. The multi-scale convolutions further encode the projected vectors into a 320-dimensional latent space (i.e., $C' = 320$). We cut the original time series into sub-sequences of length T = 402, where each sub-sequence serves as a training sample. Within each sample, the first 201 timestamps correspond to its *history* view and the subsequent 201 timestamps to its *future* view. The cosine similarity loss is optimized using stochastic gradient descent (SGD) optimizer with a learning rate of 0.001, a momentum of 0.9, and a weight decay of 0.0001. We trained 500 epochs for all datasets with a batch size of 8.

We set the predicted horizons $L \in \{24, 48, 168, 336, 720\}$ for dataset ETTh1, ETTh2, Exchange, and Weather. For dataset ETTm1 and ETTm2, we set $L \in \{24, 48, 96, 288, 672\}$. We select the best ridge regression model using the validation set and then use it to report the forecasting error on the test set. Mean-squared-error (MSE) and mean-absolute-error (MAE) are used to evaluate our results. More details about the experimental setup and training process are included in the appendix, and codes for reproducing the results will be available upon acceptance.

4.3. Results

Table 1 summarizes the average results of multivariate forecasting with five runs. Overall, our model, SimTS, outperforms all the representation learning baselines in the multivariate setting on most of the datasets by a large margin. When looking at the average performance across six datasets, SimTS outperforms TS2Vec by 18.8% (MSE) and 11.1%(MAE), TNC by 36.9% (MSE) and 16.0%(MAE), and CoST by 11.0% (MSE) and 6.8%(MAE). Additionally, when examining the performance on each dataset individually, SimTS outperforms TS2Vec on all six datasets and outperforms TNC and CoST on five out of six datasets while performing comparably or slightly worse on one of the six datasets. We believe one probable explanation is that the Exchange dataset is less stationary, and the pattern of data adjacent in time (i.e., in a neighborhood) can be discriminated from the pattern of data far away. Such neighborhood patterns can be found via TNC, which leads to better performance. On the other hand, the weather dataset is more stationary, which means CoST can use season-trend disentanglement to extract useful information and thus achieves better performance.

Although CoST and TNC perform better in some datasets, SimTS achieves overall state-of-the-art performance across all datasets. This suggests that our approach is general and robust across a wide range of time series datasets.

5. Ablation Study

In this section, we present a systematic ablation study to examine the different components and assumptions in our model. We also investigate assumptions in the baseline models to assess their influence on forecasting performance.

5.1. Backbones

First, we examine the importance of our encoder network structure design. To test the contribution of the convolutional network structure as our encoding network, we substitute the convolutional layers with the TCN (Bai et al., 2018) and LSTM (Hochreiter & Schmidhuber, 1997) networks with comparable parameter sizes. Table 2 shows the forecasting results on ETT datasets. In both univariate and multivariate forecasting, the convolutional layer in our model performs better than TCN and LSTM, demonstrating the efficiency of our encoder for encoding time series representations.

Backbones	Ours		TCN		LSTM	
	MSE	MAE	MSE	MAE	MSE	MAE
Multivariate	**0.688**	**0.601**	0.912	0.674	2.124	0.827
Univariate	**0.041**	**0.220**	0.134	0.250	1.750	1.274

Table 2. Ablation study of different backbone architectures on ETT datasets.

[1] https://github.com/laiguokun/multivariate-time-series-data
[2] https://www.bgc-jena.mpg.de/wetter/

Methods		Unsupervised Representation Learning						End-to-end Forecasting					
		Ours		TS2Vec		TNC		CoST		Informer		TCN	
	L	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	**0.377**	**0.422**	0.590	0.531	0.708	0.592	0.386	0.429	0.577	0.549	0.583	0.547
	48	**0.427**	**0.454**	0.624	0.555	0.749	0.619	0.437	0.464	0.685	0.625	0.670	0.606
	168	**0.638**	**0.577**	0.762	0.639	0.884	0.699	0.643	0.582	0.931	0.752	0.811	0.680
	336	0.815	**0.685**	0.931	0.728	1.020	0.768	**0.812**	0.679	1.128	0.873	1.132	0.815
	720	**0.956**	**0.771**	1.063	0.799	1.157	0.830	0.970	0.771	1.215	1.869	1.165	0.813
ETTh2	24	**0.336**	**0.434**	0.424	0.489	0.612	0.595	0.447	0.502	0.720	0.665	0.935	0.754
	48	**0.564**	**0.571**	0.619	0.605	0.840	0.716	0.699	0.637	1.457	1.001	1.300	0.911
	168	**1.407**	**0.926**	1.845	1.074	2.359	1.213	1.549	0.982	3.489	1.515	4.017	1.579
	336	**1.640**	**0.996**	2.194	1.197	2.782	1.349	1.749	1.042	2.723	1.340	3.460	1.456
	720	**1.878**	**1.065**	2.636	1.370	2.753	1.394	1.971	1.092	3.467	1.473	3.106	1.381
ETTm1	24	**0.232**	**0.314**	0.453	0.444	0.522	0.472	0.246	0.329	0.323	0.369	0.522	0.472
	48	**0.311**	**0.368**	0.592	0.521	0.695	0.567	0.381	0.386	0.494	0.503	0.542	0.508
	96	**0.360**	**0.402**	0.635	0.554	0.731	0.595	0.378	0.419	0.678	0.614	0.666	0.578
	288	**0.450**	**0.467**	0.693	0.597	0.818	0.649	0.472	0.486	1.056	0.786	0.991	0.735
	672	**0.612**	**0.563**	0.782	0.653	0.932	0.712	0.620	0.574	1.192	0.926	1.032	0.756
ETTm2	24	**0.108**	**0.223**	0.180	0.293	0.185	0.297	0.122	0.244	0.173	0.301	0.180	0.324
	48	**0.164**	**0.285**	0.244	0.350	0.264	0.360	0.183	0.305	0.303	0.409	0.204	0.327
	96	**0.271**	**0.376**	0.360	0.427	0.389	0.458	0.294	0.394	0.365	0.453	3.041	1.330
	288	**0.716**	**0.646**	0.723	0.639	0.920	0.788	0.723	0.652	1.047	0.804	3.162	1.337
	672	**1.600**	**0.979**	1.753	1.007	2.164	1.135	1.899	1.073	3.126	1.302	3.624	1.484
Exchange	24	**0.059**	**0.172**	0.108	0.252	0.105	0.236	0.136	0.291	0.611	0.626	2.483	1.327
	48	**0.135**	**0.265**	0.200	0.341	0.162	0.270	0.250	0.387	0.680	0.644	2.328	1.256
	168	0.713	0.635	0.412	0.492	**0.397**	**0.480**	0.924	0.762	1.097	0.825	2.372	1.279
	336	1.409	0.938	1.339	0.901	**1.008**	**0.866**	1.774	1.063	1.672	1.036	3.113	1.459
	720	**1.628**	**1.056**	2.114	1.125	1.989	1.063	2.160	1.209	2.478	1.310	3.150	1.458
Weather	24	**0.298**	**0.359**	0.308	0.364	0.320	0.373	**0.298**	0.360	0.335	0.381	0.321	0.367
	48	**0.359**	**0.410**	0.375	0.417	0.380	0.421	**0.359**	0.411	0.395	0.459	0.386	0.423
	168	**0.426**	**0.461**	0.496	0.506	0.479	0.495	0.464	0.491	0.608	0.567	0.491	0.501
	336	0.504	0.520	0.532	0.533	0.505	0.514	**0.497**	**0.517**	0.702	0.620	0.502	0.507
	720	0.535	0.542	0.567	0.558	0.543	0.547	0.533	0.542	0.831	0.731	**0.498**	**0.508**
Avg.		**0.664**	**0.562**	0.818	0.632	0.909	0.669	0.746	0.603	1.151	0.812	1.560	0.884

- The results of TS2Vec and CoST on ETTm2, Exchange, and Weather datasets are implemented by us.

Table 1. Multivariate forecasting results. The best results are highlighted in bold, and the second-best results are highlighted with an underline. L denotes the predicted horizons of datasets.The performance is measured in mean-squared error (MSE) and mean-absolute error (MAE).

5.2. Negative Samples

Negative pairs, if not constructed carefully, could depreciate the model performance in terms of representation power. TS2Vec (Yue et al., 2022) and CoST (Woo et al., 2022) use sub-sequences of other instances or various timestamps as the negative pairs for contrastive learning. However, our model SimTS outperforms them in the absence of negative pairs, implying that the selection of negative pairs in CoST and TS2Vec may be inaccurate and result in sub-optimal performance. To further demonstrate the influence of negative samples, we construct negative pairs by following SimCLR (Chen et al., 2020) to test our model result with and without negative pairs. We replace the cosine similarity loss in Equation (2) with the loss that is used in Chen et al. (2020) and Oord et al. (2018) to consider the negative pairs together with the positive pairs. Given a mini-batch $\mathcal{D} = \{X_1, X_2, ..., X_N\}$ of N samples with length T and its

Datasets	ETTh1		ETTh2		ETTm1		ETTm2		Exchange		Weather	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
SimTS	0.642	0.582	1.165	0.798	0.393	0.423	0.572	0.502	0.789	0.613	0.424	0.458
SimTS w/ neg‡	0.685	0.632	1.544	0.938	0.392	0.441	0.747	0.572	1.405	0.769	0.434	0.468

‡ With negative samples and InfoNCE loss in Equation 4.

Table 3. Ablation study of negative samples on multivariate forecasting across ETT datasets.

encoded representations $\{Z_1, Z_2, ..., Z_N\}$, we optimize:

$$\mathcal{L}^{\text{NCE}}_{\theta,\phi}(\mathcal{D}) = -\frac{1}{T}\sum_{t=1}^{T}\left[\log\frac{\exp(\hat{z}^f_t \cdot z^f_{t,+})}{\sum_{i=1}^{N}\exp(\hat{z}^f_t \cdot z^f_{t,i})}\right], \quad (4)$$

where $\hat{z}^f_{t,+} = G_\theta(z^h_t)$, and $z^f_{t,i}$ denotes latent representation of the t-th timestamp of the i-th sample from $\mathcal{D}$. The numerator calculates the similarity between predicted and encoded future representations, which is the positive pair in our framework. The denominator calculates the similarities between the negative pairs, which are the predicted future representation and encoded representations from other samples within $\mathcal{D}$. Table 3 shows the forecasting results with and without including the negative samples. In particular, it demonstrates that negative samples generally decrease performance in most of the datasets we tested. These results confirm that adding negative pairs to our proposed method leads to suboptimal performance. However, this does not mean that including negative pairs overall is not useful; it simply implies that the current approaches to constructing negative pairs are inefficient. Thus, future research should be dedicated to coming up with better ways to construct negative pairs.

5.3. Stop-Gradient Operation

In SimTS, we apply a stop-gradient operation on the *future* encoding path during the optimization. To test the effect of this operation on the overall model performance, we did an ablation study on what happens if we remove this operation, or apply it on the history encoding path instead of the *future* encoding path, see Figure 4. When we apply the stop gradient on the history encoding path, as it is shown in Figure 4c, the model optimizes the loss by pushing *future* representations Z^f towards the *future* predictions $\hat{Z}^f$. We refer to this model as RevSimTS (SimTS with reverse stop-gradient, Figure 4c). As shown in Table 4, we observe that either the removal of the stop-gradient on the *future* encoding path (Figure 4b) or moving the stop-gradient to the history encoding path causes a significant decrease in performance, supporting our argument that the stop-gradient operation in the future encoding path leads to optimal performance.

Model	SimTS		SimTS w/o SG†		RevSimTS	
	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.642**	**0.582**	0.783	0.663	0.762	0.634
ETTh2	**1.165**	**0.798**	2.940	1.490	3.128	1.449
ETTm1	**0.393**	**0.432**	0.681	0.609	0.551	0.525
ETTm2	**0.572**	**0.502**	1.315	0.863	1.186	0.796
Exchange	**0.789**	**0.613**	1.808	1.062	1.398	0.900
Weather	**0.424**	**0.458**	0.605	0.592	0.485	0.512

† SimTS without stop-gradient operation

Table 4. Ablation study of stop-gradient operation on multivariate forecasting across ETT datasets.

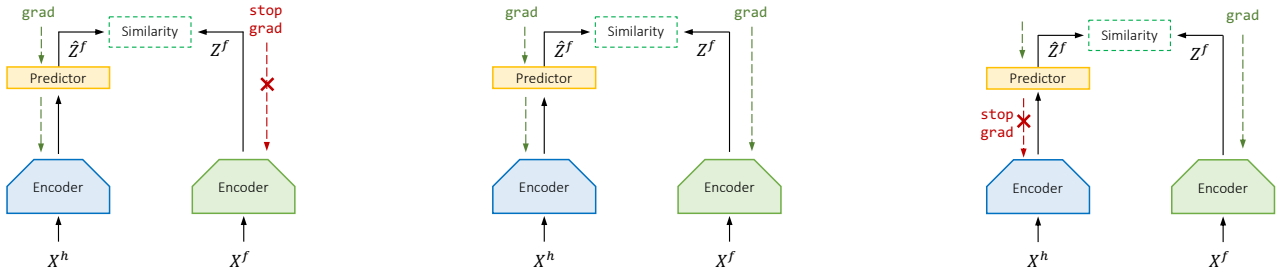
5.4. Disentanglement Assumption

Datasets	Exchange	ETTm2	ETTm1	Weather
ADF Test Stat.	-1.889	-6.225	-14.985	-26.661
CoST	0.975	0.822	0.492	0.439
CoST w/o SD†	0.899	0.754	0.466	0.440
CoST w/o aug.‡	0.865	0.986	0.493	0.462
CoST w/ mask§	1.223	0.664	1.041	0.502
Diff. w/o SD	0.076 ↑	0.068 ↑	0.026 ↑	0.001 ↓
Diff. w/o aug.	0.119 ↑	0.164 ↓	0.001 ↓	0.023 ↑
Diff. w/ mask	0.248 ↓	0.158 ↑	0.549 ↓	0.063 ↓

† Seasonal disentanglement
‡ Discard the augmentations proposed in (Woo et al., 2022)
§ Timestamp masking proposed in (Yue et al., 2022)
↑/↓ indicates performance increase/decrease

Table 5. The average multivariate forecasting results from changing the season-trend disentanglement and data augmentation modules in CoST.

To demonstrate that the season-trend disentanglement as proposed in CoST (Woo et al., 2022) may not work well for different types of datasets, especially on the less stationary data, we conduct an ablation study by removing the season disentanglement in CoST. The original season-trend disentanglement is performed by applying Fourier transform to the data and using an affine transformation to extract feature correlations in the frequency domain. We substitute this process by performing the same affine transformation on the original data without applying Fourier transform. The results are shown in Table 5, where "w/o SD" denotes CoST without the seasonal disentanglement. Besides, we adopt the



(a) SimTS (b) SimTS w/o SG (c) RevSimTS

Figure 4. Ablation study of stop-gradient operation. (a) SimTS architecture. (b) SimTS without stop-gradient operation. (c) RevSimTS with stop-gradient on the *history* encoding path.

Augmented Dick-Fuller (ADF) test statistic (Elliott et al., 1996) proposed in (Liu et al., 2022) to measure the degree of stationarity. A smaller ADF score indicates higher stationarity. We observe that seasonal disentanglement can improve the forecasting outcomes for the Weather dataset, which exhibits significant stationarity. However, the seasonal disentanglement impairs predicting ability in less stationary datasets like Exchange and ETTm2, supporting our claim that the seasonal disentanglement assumption is misleading in some datasets and lacks generality.

5.5. Data Augmentation for Constructing Views

Data augmentation is a common method to generate positive pairs in contrastive learning. However, current augmentation methods for time series may impair the performance of forecasting. We conduct ablation studies to demonstrate the influences of data augmentations. CoST uses three types of data augmentation: scaling, shifting, and jittering. On the other hand, TS2Vec randomly masks timestamps in a sample to construct views. Therefore, we implement two ablation experiments for CoST: (1) eliminating data augmentation and (2) adding random masks. Table 5 shows the results of the two experiments, where "w/o aug" denotes CoST without its original augmentation methods and "w/mask" denotes CoST using random masks as augmentation. Our experiments show that the original data augmentation in CoST can potentially result in lower performances, and adding random masks impairs performances for most datasets. These findings do not imply that data augmentation is not effective in general; rather, they demonstrate that finding efficient augmentation techniques applicable to various time series is challenging, and better methods for augmenting time series data need to be developed.

6. Conclusion

This paper proposes SimTS, a simple representation learning framework based on contrastive learning that does not require negative pairs. We conducted an extensive study to test our proposed model and compared it to other existing representation learning models for time series forecasting. Our general aim was to challenge the assumptions and components that are widely used in these models. Our study reveals that current representation learning methods are not universally applicable to different types of time series data. Some of the components used in these models might be unnecessary and can even negatively impact performance in some cases. This means that existing models based on contrastive learning for time series forecasting are highly dependent on the specific dataset being used, and careful consideration is necessary when deploying them.

Our proposed model, however, addresses some of the limitations by providing a simplified and robust contrastive learning model achieving better performance across different datasets compared to state-of-the-art methods. Moving forward, we plan to extend our framework to handle more challenging data such as irregular time series and explore efficient data augmentation methods for time series forecasting.

7. Acknowledgements

This work is supported by the Swiss National Science Foundation (project 201184).

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.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 1993.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Chen, X. and He, K. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021.

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.

Elliott, G., Rothenberg, T. J., and Stock, J. H. Efficient tests for an autoregressive unit root. *Econometrica*, 64: 813–836, 1996.

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

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Han, Z., Liu, Y., Zhao, J., and Wang, W. Real time prediction for converter gas tank levels based on multi-output least square support vector regressor. *Control Engineering Practice*, 20(12):1400–1409, 2012.

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*, pp. 9729–9738, 2020.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hyvarinen, A. and Morioka, H. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *Advances in neural information processing systems*, 29, 2016.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

Kiyasseh, D., Zhu, T., and Clifton, D. A. Clocs: Contrastive learning of cardiac signals across space, time, and patients. In *International Conference on Machine Learning*. PMLR, 2021.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Liu, C., Hoi, S. C., Zhao, P., and Sun, J. Online arima algorithms for time series prediction. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

Liu, Y., Wu, H., Wang, J., and Long, M. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Nonnenmacher, M. T., Oldenburg, L., Steinwart, I., and Reeb, D. Utilizing expert features for contrastive learning of time-series representations. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

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

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.

Sezer, O. B., Gudelek, M. U., and Ozbayoglu, A. M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

Shumway, R. H., Stoffer, D. S., and Stoffer, D. S. *Time series analysis and its applications*, volume 3. Springer, 2000.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 2020.

Tian, Y., Chen, X., and Ganguli, S. Understanding self-supervised learning dynamics without contrastive pairs. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Tonekaboni, S., Eytan, D., and Goldenberg, A. Unsupervised representation learning for time series with temporal neighborhood coding. In *International Conference on Learning Representations*, 2021.

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 (NeurIPS)*, 30, 2017.

Wang, Z., Xu, X., Zhang, W., Trajcevski, G., Zhong, T., and Zhou, F. Learning latent seasonal-trend representations for time series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 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 (ICLR)*, 2022.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Yang, L. and Hong, S. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2022.

Ye, L. and Keogh, E. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 947–956, 2009.

Yèche, H., Dresdner, G., Locatello, F., Hüser, M., and Rätsch, G. Neighborhood contrastive learning applied to online patient moninring. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

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*, 2022.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.

Zhang, C., Zhang, K., Zhang, C., Pham, T. X., Yoo, C. D., and Kweon, I. S. How does simsiam avoid collapse without negative samples? a unified understanding with self-supervised contrastive learning. *arXiv preprint arXiv:2203.16262*, 2022a.

Zhang, X., Zhao, Z., Tsiligkaridis, T., and Zitnik, M. Self-supervised contrastive pre-training for time series via time-frequency consistency. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b.

Zheng, J. and Huang, M. Traffic flow forecast through time series analysis based on deep learning. *IEEE Access*, 8: 82562–82570, 2020.

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*, pp. 11106–11115, 2021.

A. Summary of Time Series Forecasting Datasets

Dataset	Variable Number	Sampling Frequency	Total Observations	ADF Test Statistic
ETTh1	7	1 Hour	17,420	-5.909
ETTh2	7	1 Hour	17,420	-4.136
ETTm1	7	15 Minutes	69,680	-14.985
ETTm2	7	15 Minutes	69,680	-6.225
Exchange	8	1 Day	7,588	-1.889
Weather	21	10 Minutes	52,695	-26.661

Table 6. Summary of datasets. Smaller ADF test statistic indicates a more stationary dataset.

B. Details on baselines

The seven baselines' descriptions and implementations are listed below. We reproduce the results of CoST, TS2Vec, TNC, and Informer for dataset ETTm2, Exchange and Weather. Other results are taken from Woo et al. (2022) and Wang et al. (2022). Unless otherwise stated, we employ the parameters specified in the respective papers.

CoST (Woo et al., 2022): CoST performs season-trend disentanglement to learn seasonal and trend representations separately by using Fourier Transform. The final representation for forecasting is the concatenation of the seasonal and trend representation. We run their code from https://github.com/salesforce/CoST.

TS2Vec (Yue et al., 2022): TS2Vec designs a hierarchical contrastive learning framework to learn a universal time series representation. It employs timestep masks as the data augmentation and temporal convolutions to encode the latent representations. We reproduce their experiments from their publicly available code: https://github.com/yuezhihan/ts2vec

TNC (Tonekaboni et al., 2021): TNC is an unsupervised representation learning method that makes sure the latent representations from a neighborhood are distinguishable from representations outsides the neighborhood. We use their open source code from https://github.com/sanatonek/TNC_representation_learning. Following the setup in TS2Vec, we use the casual TCN encoder proposed in TS2Vec to replace the original encoder in TNC.

Informer (Zhou et al., 2021): Informer is designed based on the transformer for long sequence time series forecasting. It consists of three major components: a ProbSparse self-attention mechanism, a self-attention distilling mechanism, and a generative style decoder. We use their code from https://github.com/zhouhaoyi/Informer2020

TCN (Oord et al., 2016): TCN proposes dilated convolutions for time series data. A stack of ten residual blocks with a hidden size of 64 is added to the encoder in TS2Vec. Their public source code can be achieved at https://github.com/locuslab/TCN.

C. Results for Univariate Setting

Methods		Unsupervised Representation Learning						End-to-end Forecasting					
		Ours		TS2Vec		TNC		CoST		Informer		TCN	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	**0.036**	**0.143**	0.039	0.151	0.057	0.184	0.040	0.152	0.098	0.147	0.104	0.254
	48	**0.054**	**0.176**	0.062	0.189	0.094	0.239	0.060	0.186	0.158	0.319	0.206	0.366
	168	**0.084**	**0.216**	0.142	0.291	0.171	0.329	0.097	0.236	0.183	0.346	0.462	0.586
	336	**0.100**	**0.239**	0.160	0.316	0.179	0.345	0.112	0.258	0.222	0.387	0.422	0.564
	720	**0.126**	**0.277**	0.179	0.345	0.235	0.408	0.148	0.306	0.269	0.435	0.438	0.578
ETTh2	24	**0.077**	**0.206**	0.097	0.230	0.097	0.238	0.079	0.207	0.093	0.240	0.109	0.251
	48	**0.116**	**0.259**	0.124	0.274	0.131	0.281	0.118	0.259	0.155	0.314	0.147	0.302
	168	0.191	0.340	0.198	0.355	0.197	0.354	**0.189**	**0.339**	0.232	0.389	0.209	0.366
	336	**0.199**	**0.354**	0.205	0.364	0.207	0.366	0.206	0.360	0.263	0.417	0.237	0.391
	720	**0.212**	**0.370**	0.208	0.371	0.207	0.370	0.214	0.371	0.277	0.431	0.200	0.367
ETTm1	24	**0.013**	**0.084**	0.016	0.093	0.019	0.103	0.015	0.088	0.030	0.137	0.027	0.127
	48	**0.024**	**0.112**	0.028	0.126	0.045	0.162	0.025	0.117	0.069	0.203	0.040	0.154
	96	**0.041**	**0.143**	0.045	0.162	0.054	0.178	0.038	0.147	0.194	0.372	0.097	0.246
	288	**0.098**	**0.207**	0.095	0.235	0.142	0.290	0.077	0.209	0.401	0.544	0.305	0.455
	672	0.117	**0.242**	0.142	0.290	0.136	0.290	**0.113**	0.257	0.277	0.431	0.200	0.367
ETTm2	24	**0.022**	**0.099**	0.038	0.139	0.045	0.151	0.027	0.112	0.036	0.141	0.048	0.153
	48	**0.045**	**0.149**	0.069	0.194	0.080	0.201	0.054	0.159	0.069	0.200	0.063	0.191
	96	**0.068**	**0.189**	0.089	0.225	0.094	0.229	0.072	0.196	0.095	0.240	0.129	0.265
	288	0.160	0.272	0.161	0.306	0.155	0.309	**0.153**	**0.307**	0.211	0.367	0.208	0.352
	672	0.249	0.334	0.201	0.351	0.197	0.352	**0.183**	**0.329**	0.267	0.417	0.222	0.377
Exchange	24	**0.027**	**0.128**	0.033	0.142	0.082	0.227	0.028	0.128	0.103	0.262	-	-
	48	**0.049**	**0.169**	0.059	0.191	0.116	0.268	0.048	0.169	0.121	0.283	-	-
	168	**0.158**	**0.314**	0.180	0.340	0.275	0.411	0.161	0.319	0.168	0.337	-	-
	336	**0.382**	**0.488**	0.465	0.533	0.579	0.582	0.399	0.497	1.672	1.036	-	-
	720	1.600	1.016	**1.357**	**0.931**	1.570	1.024	1.639	1.044	2.478	1.310	-	-
Weather	24	0.098	0.214	**0.096**	0.215	0.102	0.221	**0.096**	**0.213**	0.117	0.251	0.109	0.217
	48	**0.136**	**0.260**	0.140	0.264	0.139	0.264	0.138	0.262	0.178	0.318	0.143	0.269
	168	**0.120**	**0.328**	0.207	0.335	0.198	0.328	0.207	0.334	0.266	0.398	0.188	0.319
	336	0.221	0.349	0.231	0.360	0.215	0.347	0.230	0.356	0.197	0.416	**0.192**	**0.320**
	720	0.235	0.365	0.233	0.365	0.219	0.353	0.242	0.370	0.359	0.466	**0.198**	**0.329**
Avg.		**0.169**	**0.268**	0.176	0.289	0.201	0.313	0.174	0.176	0.309	0.385	-	-

Table 7. Univariate forecasting results of ETT datasets. The best results are highlighted in bold. L denotes the predicted horizons of datasets. The performances are measured in mean-squared error (MSE) and mean-absolute error (MAE).