Simple Contrastive Representation Learning for Time Series Forecasting

Xiaochen Zheng[13*†], **Xingyu Chen**[2*], **Manuel Schürch**[13],
Amina Mollaysa[13], **Ahmed Allam**[1], **Michael Krauthammer**[13],
[*]equal contribution [†]corresponding author
[1]University of Zurich [2]ETH Zurich [3]ETH AI Center
xiaochen.zheng@uzh.ch, xingyuchen@ethz.ch

Abstract

Contrastive learning methods have shown an impressive ability to learn meaningful representations for image and 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 outcomes from the historical context. 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 exclusively uses positive pairs and does not depend on negative pairs or specific characteristics of a given time series. In addition, we show the shortcomings of the common contrastive learning frameworks 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 signal processing has recently experienced significant progress [36, 5]. In particular, time series forecasting plays an important role in addressing several practical real-world applications [14, 6]. Among them, self-supervised learning approaches such as contrastive learning [33, 34, 30, 14] have shown promise in exploiting these applications and have continually outperformed supervised approaches [1, 22, 38] in time series forecasting tasks.

However, current instance discrimination [31] based contrastive mechanisms may not be sufficient for accurate forecasting. In contrastive time series classification, the model aims to learn representations that can differentiate between time series instances. The resulting representations contain information that can discriminate well between different time series instances, making them informative for *classification*. Meanwhile, in time series forecasting, the objective is to predict future values based on the past rather than discriminating between instances. The representations learned through instance discrimination could not capture the necessary information for accurate forecasting. Other features, such as the trend, seasonality, and dependencies within the time series, can be more relevant for forecasting.

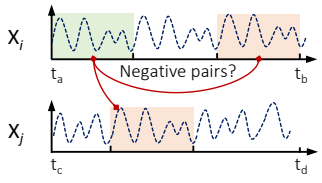


Figure 1: Problems with selecting negative pairs based on methods proposed in [33, 34, 30] when cross-instance and cross-time repeated patterns exist.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Moreover, contrastive learning relies on semantic preserving data augmentations to create positive and negative pairs, but finding suitable augmentations for time series forecasting is challenging [18]. As shown in Figure 1, some existing methods [34, 14, 33, 26] assume that the similarity between segments decreases with increasing time lag, which may not hold for all time series. This can result in selecting inappropriate negative pairs [24, 19], leading to false repulsion. Other recent approaches [30, 28, 32] rely on assumptions about time series decomposition (e.g., trends and seasonality), which may not generalize well to diverse forecasting datasets.

To address these limitations, we propose a ***Sim**ple Representation Learning Framework for **T**ime **S**eries Forecasting* (**SimTS**), which is inspired by predictive coding [21, 9]. In particular, we build upon a siamese network structure [4, 37] and propose key refinements that enable better prediction performance with a simpler model structure compared to state-of-the-art methods. Specifically, inspired by [4, 9, 25], we propose to not use negative pairs. Moreover, we demonstrate that maximizing the shared information between the representations of *history* and *future* time windows is important for forecasting tasks with representation learning. In our proposed model, we explicitly impose a constraint that the learned representation of the history should encode as much information as possible by predicting the latent representation of the future. This mechanism simplifies several existing approaches and leads to state-of-the-art forecasting results, as thoroughly demonstrated in this paper.

2 Methods

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.

SimTS aims to learn time series representations by maximizing the similarity between predicted and encoded latent features for each timestamp. One of our main aims is to design a framework capable of conforming to inherent characteristics of forecasting tasks. 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. In SimTS, we simply apply multiple single-layer CNNs (mslCNN) with different kernel sizes as the encoder to learn multiscale temporal information.

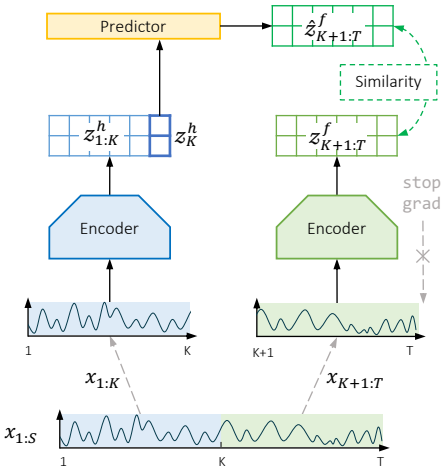


Figure 2: Illustration of our proposed SimTS.

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

Methods	Ours		TS2Vec		TNC		CoST		InfoTS*	
Accepted by			AAAI 2022		ICLR 2021		ICLR 2022		AAAI 2023	
Dataset	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.643**	**0.582**	0.794	0.650	0.904	0.702	0.650	0.585	0.784	0.649
ETTh2	**1.165**	**0.798**	1.544	0.947	1.869	1.053	1.283	0.851	1.474	0.914
ETTm1	**0.393**	**0.423**	0.631	0.554	0.740	0.599	0.419	0.439	0.568	0.521
ETTm2	**0.572**	**0.502**	0.652	0.543	0.784	0.608	0.644	0.534	0.686	0.575
Exchange	0.789	0.613	0.835	0.622	**0.732**	**0.583**	1.043	0.742	0.928	0.654
Weather	**0.424**	**0.458**	0.456	0.476	0.445	0.470	0.430	0.459	1.220	0.837
Avg.	**0.664**	**0.562**	0.818	0.632	0.909	0.669	0.746	0.603	0.954	0.692

- We implement TS2Vec and CoST on ETTm2, Exchange, and Weather datasets.
* We reproduce InfoTS results using the code available at `openreview.net/forum?id=kxARp2zoqAk`.

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

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 $Z^f = F_\theta(X^f) = [z^f_{K+1}, z^f_{K+2}, ..., z^f_T] \in \mathbb{R}^{C' \times (T-K)}$. Inspired by [35], 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 the latent space without introducing 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: $Sim(\hat{Z}^f, Z^f) = -\frac{1}{T-K} \sum_{i=K+1}^{T} \frac{\hat{z}^f_i}{\|\hat{z}^f_i\|_2} \cdot \frac{z^f_i}{\|z^f_i\|_2}$, where $\|\cdot\|_2$ is l_2-norm and $Sim(\cdot)$ is the average cosine similarity of all time steps.

Empirically, we witness better performance when applying a stop-gradient (sg) operation [4, 37] to the *future* encoding path in our model. With stop-gradient, the loss is:

$$\mathcal{L}_{\theta,\phi}(X^h, X^f) = Sim\left(G_\phi\left(F_\theta(X^h)\right), F_{\mathtt{sg}(\theta)}(X^f)\right) = Sim(\hat{Z}^f, \mathtt{sg}(Z^f)) \tag{1}$$

The loss in definition (1) is for one sample $X = [X^h, X^f]$ and can be adapted for a mini-batch $\mathcal{D} = \{X^h_i, X^f_i\}_{i \in [1:N]}$, that is,

$$\mathcal{L}_{\theta,\phi}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\theta,\phi}(X^h_i, X^f_i), \tag{2}$$

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

3 Results

Table 1 shows that for multivariate time series forecasting. The results is averaged over five different forecasting lengths for each dataset. The details of results on each forecasting length are shown in Appendix. SimTS consistently outperforms all representation learning baselines, surpassing the second-best method, CoST, by 8.2% (MSE) and 4.1% (MAE) across six datasets. Additionally, when examining the performance on each dataset individually, SimTS performs slightly worse on the Exchange dataset.

Please note that we have not compared our results to CPC [21] since (1) CPC is not specifically designed for time series forecasting, and (2) we have outperformed BTSF [32] while BTSF reports that their model can outperform CPC.

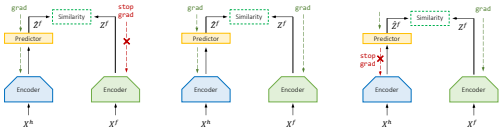
4 Ablation Study

Negative Samples Negative pairs, if not constructed carefully, could depreciate the model performance in terms of representation power. To further demonstrate the influence of negative samples, we construct negative pairs by following SimCLR [3] to test our model result with and without negative pairs. We replace the cosine similarity loss in Equation 1 with the loss that is used in [3] and [21] to consider the negative pairs together with the positive pairs. Table 2 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 for constructing negative pairs are inefficient. Thus, future research should be dedicated to coming up with better ways to construct negative pairs.

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
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

Table 2: Ablation study of negative samples on multivariate forecasting across ETT datasets.

Stop-Gradient Operation To test the effect of the stop-gradient operation on the overall model performance, we construct an ablation study reported in [37], as illustrated in Figure 3. As shown in Table 4, we observe that either the removal of the stop-gradient on the *future* encoding path (Figure 3b) 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.



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

Figure 3: 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.

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**	**0.423**	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

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

5 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 the performance of existing models based on contrastive learning for time series forecasting depends highly on the datasets they are applied to, 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.

References

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[2] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607. PMLR, 2020.

[4] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

[5] Ting Dang, Antoni Dimitriadis, Jingyao Wu, Vidhyasaharan Sethu, and Eliathamby Ambikairajah. Constrained dynamical neural ode for time series modelling: A case study on continuous emotion prediction. In *ICASSP*, 2023.

[6] Dazhao Du, Bing Su, and Zhewei Wei. Preformer: predictive transformer with multi-scale segment-wise correlations for long-term time series forecasting. In *ICASSP*, 2023.

[7] Graham Elliott, Thomas J. Rothenberg, and James H. Stock. Efficient tests for an autoregressive unit root. *Econometrica*, 64:813–836, 1996.

[8] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *NeurIPS*, 32, 2019.

[9] Jean-Bastien Grill, Florian Strub, Florent Altché, et al. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*, 2020.

[10] Zhongyang Han, Ying Liu, Jun Zhao, and Wei Wang. 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.

[11] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Aapo Hyvarinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *NeurIPS*, 29, 2016.

[14] Dani Kiyasseh, Tingting Zhu, and David A Clifton. Clocs: Contrastive learning of cardiac signals across space, time, and patients. In *ICML*. PMLR, 2021.

[15] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *ACM SIGIR*, pages 95–104, 2018.

[16] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *NeurIPS*, 32, 2019.

[17] Chenghao Liu, Steven CH Hoi, Peilin Zhao, and Jianling Sun. Online arima algorithms for time series prediction. In *AAAI*, 2016.

[18] Dongsheng Luo, Wei Cheng, Yingheng Wang, et al. Time series contrastive learning with information-aware augmentations. In *AAAI*, 2023.

[19] Manuel T Nonnenmacher, Lukas Oldenburg, Ingo Steinwart, and David Reeb. Utilizing expert features for contrastive learning of time-series representations. In *ICML*, 2022.

[20] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[21] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[22] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2020.

[23] Robert H Shumway, David S Stoffer, and David S Stoffer. *Time series analysis and its applications*, volume 3. Springer, 2000.

[24] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *NeurIPS*, 2020.

[25] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *ICML*, 2021.

[26] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. In *ICLR*, 2021.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[28] Zhiyuan Wang, Xovee Xu, Weifeng Zhang, Goce Trajcevski, Ting Zhong, and Fan Zhou. Learning latent seasonal-trend representations for time series forecasting. In *NeurIPS*, 2022.

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

[30] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *ICLR*, 2022.

[31] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

[32] Ling Yang and Shenda Hong. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *ICML*, 2022.

[33] Hugo Yèche, Gideon Dresdner, Francesco Locatello, Matthias Hüser, and Gunnar Rätsch. Neighborhood contrastive learning applied to online patient moninring. In *ICML*, 2021.

[34] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *AAAI*, 2022.

[35] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.

[36] Jun Zhan, Siqi Wang, Xiandong Ma, Chengkun Wu, Canqun Yang, Detian Zeng, and Shilin Wang. Stgat-mad: Spatial-temporal graph attention network for multivariate time series anomaly detection. In *ICASSP*, 2022.

[37] Chaoning Zhang, Kang Zhang, Chenshuang Zhang, Trung X Pham, Chang D Yoo, and In So Kweon. How does simsiam avoid collapse without negative samples? a unified understanding with self-supervised contrastive learning. *arXiv preprint arXiv:2203.16262*, 2022.

[38] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, 2021.

6 Supplementary Material

6.1 Datasets

Our experiments are carried out on six real-world public benchmark datasets. **Electricity Transformer Temperature (ETT)** [38] measures long-term deployment of electric power. It consists of two hourly-sampled datasets (ETTh) and two 15-minute-sampled datasets (ETTm), collected for two years and from two different Chinese provinces. **Exchange-Rate**[1] [15] contains the daily exchange rates of eight foreign countries from 1990 to 2016. **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.

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 3: Summary of datasets. Smaller ADF test statistic indicates a more stationary dataset.

6.2 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 [30] and [28]. Unless otherwise stated, we employ the parameters specified in the respective papers.

CoST [30]: CoST performs season-trend disentanglement to learn seasonal and trend representations separately by using the 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 [34]: 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 [26]: TNC is an unsupervised representation learning method that makes sure the latent representations from a neighborhood are distinguishable from representations outside 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 [38]: 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 [20]: 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`.

6.3 Experimental setup

To keep a fair comparison, we follow the 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

[1] https://github.com/laiguokun/multivariate-time-series-data

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

the learned latent representation for forecasting, i.e., predicting future L time steps. 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.

7 Multiple Single-layer CNN (mslCNN)

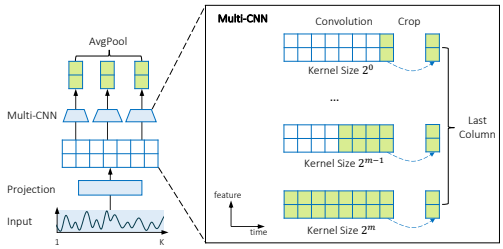


Figure 5: Multi-scale encoder. Composed of a projection layer and a set of parallel 1d convolutions with kernel size 2^i, for $i \in \{0, 1, ..., m\}$. An averaged pooling layer is added on the top of convolutions.

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

Figure 5 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 [34, 30, 28]. 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 $\hat{Z}_{(0)}, \hat{Z}_{(1)}, ..., \hat{Z}_{(m)}$.

8 Ablation Studies

8.1 Disentanglement Assumption

To demonstrate that the season-trend disentanglement as proposed in CoST [30] may not work well for different types of datasets, especially on the less stationary data, we conduct an ablation study by

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 [30]
§ Timestamp masking proposed in [34]
↑/↓ indicates performance increase/decrease

Table 4: The average multivariate forecasting results when changing the season-trend disentanglement and data augmentation modules in CoST.

Backbones	mslCNN		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 5: Ablation study of different backbone architectures on ETT datasets.

removing the season disentanglement in CoST. The results are shown in Table 4. Besides, we adopt the Augmented Dick-Fuller (ADF) test statistic [7] 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.

8.2 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 4 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.

8.3 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 [1] and LSTM [12] networks with comparable parameter sizes. Table 5 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.

9 Full Results for Univariate Setting

Methods		Ours		TS2Vec		TNC		CoST		BTSF		InfoTS	
Accepted by				AAAI 2022		ICLR 2021		ICLR 2022		ICML 2022		AAAI 2023	
	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 6: 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).

10 Full Results for Multivariate Setting

Methods		Ours		TS2Vec		TNC		CoST		InfoTS*	
Accepted by				AAAI 2022		ICLR 2021		ICLR 2022		AAAI 2023	
	L	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	0.564	0.520
	48	**0.427**	**0.454**	0.624	0.555	0.749	0.619	0.437	0.464	0.607	0.553
	168	**0.638**	**0.577**	0.762	0.639	0.884	0.699	0.643	0.582	0.746	0.638
	336	0.815	**0.685**	0.931	0.728	1.020	0.768	**0.812**	0.679	0.904	0.722
	720	**0.956**	**0.771**	1.063	0.799	1.157	0.830	0.970	0.771	1.098	0.811
ETTh2	24	**0.336**	**0.434**	0.424	0.489	0.612	0.595	0.447	0.502	0.383	0.462
	48	**0.564**	**0.571**	0.619	0.605	0.840	0.716	0.699	0.637	0.567	0.582
	168	**1.407**	**0.926**	1.845	1.074	2.359	1.213	1.549	0.982	1.789	1.048
	336	**1.640**	**0.996**	2.194	1.197	2.782	1.349	1.749	1.042	2.120	1.161
	720	**1.878**	**1.065**	2.636	1.370	2.753	1.394	1.971	1.092	2.511	1.316
ETTm1	24	**0.232**	**0.314**	0.453	0.444	0.522	0.472	0.246	0.329	0.391	0.408
	48	**0.311**	**0.368**	0.592	0.521	0.695	0.567	0.381	0.386	0.503	0.475
	96	**0.360**	**0.402**	0.635	0.554	0.731	0.595	0.378	0.419	0.537	0.503
	288	**0.450**	**0.467**	0.693	0.597	0.818	0.649	0.472	0.486	0.653	0.579
	672	**0.612**	**0.563**	0.782	0.653	0.932	0.712	0.620	0.574	0.757	0.642
ETTm2	24	**0.108**	**0.223**	0.180	0.293	0.185	0.297	0.122	0.244	0.213	0.330
	48	**0.164**	**0.285**	0.244	0.350	0.264	0.360	0.183	0.305	0.283	0.392
	96	**0.271**	**0.376**	0.360	0.427	0.389	0.458	0.294	0.394	0.360	0.449
	288	**0.716**	**0.646**	0.723	0.639	0.920	0.788	0.723	0.652	0.830	0.700
	672	**1.600**	**0.979**	1.753	1.007	2.164	1.135	1.899	1.073	1.745	1.006
Exchange	24	**0.059**	**0.172**	0.108	0.252	0.105	0.236	0.136	0.291	0.085	0.216
	48	**0.135**	**0.265**	0.200	0.341	0.162	0.270	0.250	0.387	0.176	0.310
	168	0.713	0.635	0.412	0.492	**0.397**	**0.480**	0.924	0.762	0.718	0.645
	336	1.409	0.938	1.339	0.901	**1.008**	**0.866**	1.774	1.063	1.481	0.957
	720	**1.628**	**1.056**	2.114	1.125	1.989	1.063	2.160	1.209	2.179	1.142
Weather	24	**0.298**	**0.359**	0.308	0.364	0.320	0.373	**0.298**	0.360	0.217	0.277
	48	**0.359**	**0.410**	0.375	0.417	0.380	0.421	**0.359**	0.411	0.466	0.511
	168	**0.426**	**0.461**	0.496	0.506	0.479	0.495	0.464	0.491	1.229	0.891
	336	0.504	0.520	0.532	0.533	0.505	0.514	**0.497**	**0.491**	1.880	1.134
	720	0.535	0.542	0.567	0.558	0.543	0.547	**0.533**	**0.542**	2.631	1.373
Avg.		**0.664**	**0.562**	0.818	0.632	0.909	0.669	0.746	0.603	0.954	0.692

- We implement TS2Vec and CoST on ETTm2, Exchange, and Weather datasets.

* We reproduce InfoTS results using the code available at `openreview.net/forum?id=kxARp2zoqAk`.

Table 7: 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).

11 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 [17], SVM [10], and VAR [2], have been outperformed on many datasets by deep learning models, including RNN [29], CNN [1] and transformers [27]. TCN [1] introduces dilated convolutions [20] 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 [16] suggests the *LogSparse* attention; Informer [38] 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 [11, 3] with InfoNCE loss [21]. To get reliable time-series representations, several

approaches have been investigated. Some studies focus on formulating time segments as contrastive pairs: ICA [13] investigates non-stationarity in temporal data to find a representation that allows optimal time segment discrimination; TNC [26] 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, [8] learns scalable representations for various time series lengths using contrasting positive, negative, and reference pairs with an innovative triplet loss. TS2Vec [34] 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 [23], there is growing research that uses time series decomposition in unsupervised learning. CoST [30] encodes disentangled trend and seasonal representations using contrastive learning. BTSF [32] 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.