Self-Supervised Time Series Representation Learning with Temporal-Instance Similarity Distillation

Anonymous Authors[1]

Abstract

We propose a self-supervised method for pre-training universal time series representations in which we learn contrastive representations using similarity distillation along the temporal and instance dimensions. We analyze the effectiveness of both dimensions, and evaluate our pre-trained representations on three downstream tasks: time series classification, anomaly detection, and forecasting.

1. Introduction

Large-scale pre-trained models provide the initial foundation in many real-world machine learning systems, in particular in the domains of computer vision and natural language processing. Despite the wide range of applications in healthcare, finance, transportation, energy, etc., the development of large-scale pre-trained models for time series remains under-explored in the machine learning community. An important first step towards building such a model is the self-supervised learning of a universal representation for time series. We present a novel self-supervised representation learning method for time series in which we leverage similarity distillation (Tejankar et al., 2021; Zheng et al., 2021) as an alternative source of self-supervision to traditional negative-positive contrastive pairs. In particular, we propose to learn contrastive representations using similarity distillation along the temporal and instance dimensions. We systematically analyze the effectiveness of distillation in both dimensions, as well as the benefit of hierarchical pooling. We evaluate the performance of our pre-trained time series representations on three downstream tasks: time series classification, anomaly detection, and forecasting. Our code is available at X.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the First Workshop of Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022. Do not distribute.

2. Related Work

Self-supervised techniques can be used for learning representations without requiring explicit labels. The pre-trained representations can afterwards be fine-tuned with fewer labelled data for different downstream tasks. For time series representation learning, pre-training approaches have shifted from simpler approaches such as using a sequence-to-sequence encoder-decoder architecture (Malhotra et al., 2017) to more advanced techniques, such as using either pre-text tasks (e.g., learning the masked values (Zerveas et al., 2021)) or contrastive learning on different augmentations of the input series (Eldele et al., 2021; Franceschi et al., 2019; Tonekaboni et al., 2021; Yue et al., 2022). Contrastive methods have been empirically shown to have a better performance (Yue et al., 2022; Tonekaboni et al., 2021; Eldele et al., 2021) and they are trained by *augmenting* every batch and taking the augmentations of the same input as a *positive* pair and augmentations of different inputs as *negative* pairs. The contrastive loss brings the representations of the elements in positive pairs closer to each other than the elements that form a negative pair.

However, contrastive methods for self-supervised representation learning rely on the assumption that the augmentation of a given sample will generate a negative pair with other samples in the batch. This assumption is not always valid: there could be samples of the same class in the current batch which would mean that not all the assumed negative samples are truly negative. To address this issue, instead of using positive and negative pairs with contrastive learning, we can use knowledge distillation based approaches, in which a student network is trained to produce the same similarity PDF as a teacher network (with momentum-updated weights) between the current elements in the batch and a set of anchors. This is the approach taken in ISD (Tejankar et al., 2021) and ReSSL (Zheng et al., 2021). While this approach has been used in the computer vision domain, to the best of our knowledge, it has never been used for pre-training time series representations.

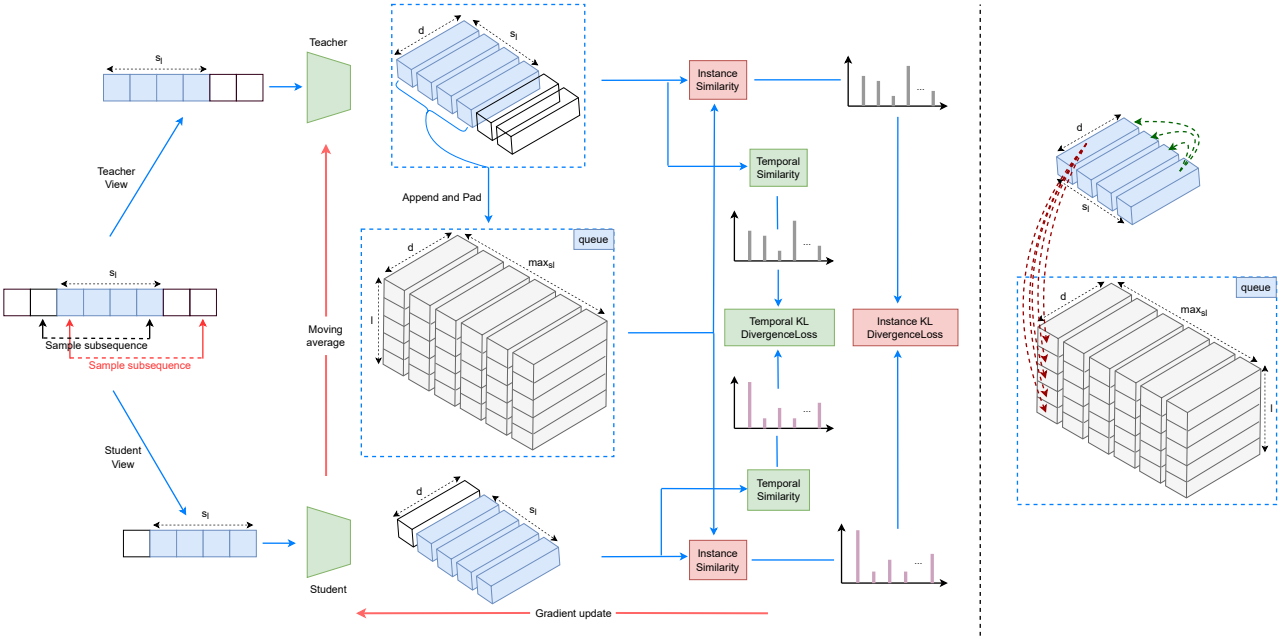

Figure 1. Overview of our proposed model architecture. Red dotted arrows show how similarities between different instances are computed, while the green arrows show how they are computed across the temporal dimension. The arrows are only shown for a few instances to avoid clutter. The overall training loss is a combination of instance and temporal losses.

3. Method

3.1. Problem Definition

We propose to learn a non-linear embedding function for a set of time series $\mathcal{X} = \{x^1, x^2, ..., x^N\}$ of size N that maps each time series x^i with T_i timestamps, to its best describing representation $\mathbf{r}^i = \{\mathbf{r}^i_1, \mathbf{r}^i_2, ..., \mathbf{r}^i_{T_i}\}$. Each $\mathbf{r}^i_j \in \mathbb{R}^d$ is the representation of time series i at timestamp j.

3.2. Model Architecture

Our overall architecture can be viewed in Figure 1. Inspired by the recent approaches in self-supervised learning (Tejankar et al., 2021; Zheng et al., 2021), we propose a student-teacher framework that uses similarity distillation to learn time series representations in a self-supervised manner. We incorporate the same augmentation technique as the state-of-the-art TS2Vec method (Yue et al., 2022), i.e., we sample two overlapping subsequences from the same sequence. The two subsequences are applied to a teacher encoder and a student encoder. Student and teacher encoders follow TS2Vec, consisting of three components: an input projection layer, a timestamp masking module, and a dilated CNN module. Gradients of the network are only propagated through the student encoder while the teacher encoder is the moving average of the student encoder to avoid collapse (Tejankar et al., 2021; Zheng et al., 2021; Chen & He, 2021).

While similarity distillation has been used in computer vision, to the best of our knowledge, we are the first to leverage this framework for time series data. As shown in Figure 1, applying the student and teacher encoders to the subsequences results in $s_l \times d$ matrices for the student and teacher representations in the overlapping region, where s_l is the length of the overlap (the representations of the non-overlapping regions are ignored in further processing). Similar to (Tejankar et al., 2021) and (Zheng et al., 2021), we use a memory buffer, implemented as a queue, to store a set of anchor sequences. For each input sequence, the teacher representations in the overlapping region are appended to the memory buffer, forming a $l \times \max s_l \times d$ matrix of anchor representations, where l is length of the buffer and $\max s_l$ is the maximum overlap length. Zero padding is applied where necessary. To learn an effective representation for time series, we want to capture the relationship between the events at various timestamps within the same sequence (temporal objective) as well as the relationship across different sequences (instance objective).

Let $\mathbf{s}_j$ denote the student representation of an augmented sequence at temporal position j (illustrated as a single d-dimensional slice at the bottom of Figure 1). The temporal loss is computed as follows. First, we contrast $\mathbf{s}_j$ with the other student representations of the same augmented sequence at all other temporal positions (green dotted arrows): given a similarity function sim, such as cosine similarity,

we obtain the s_l-dimensional probability distribution

$$\mathbf{p}_{s,j}^{\text{temp}}(k) = \frac{\exp(\text{sim}(\mathbf{s}_j, \mathbf{s}_k)/\tau)}{\sum_{m=1}^{s_l} \exp(\text{sim}(\mathbf{s}_j, \mathbf{s}_m)/\tau)}, \quad (1)$$

where τ is a temperature hyperparameter. Analogously, with $\mathbf{t}_j$ denoting the teacher representation of the augmented sequence at temporal position j (illustrated as a single d-dimensional slice at the top of Figure 1), we compute

$$\mathbf{p}_{t,j}^{\text{temp}}(k) = \frac{\exp(\text{sim}(\mathbf{t}_j, \mathbf{t}_k)/\tau)}{\sum_{m=1}^{s_l} \exp(\text{sim}(\mathbf{t}_j, \mathbf{t}_m)/\tau)}. \quad (2)$$

The temporal loss is obtained by summing the KL divergences $KL(\mathbf{p}_{t,j}^{\text{temp}}, \mathbf{p}_{s,j}^{\text{temp}})$ over all temporal positions:

$$\mathcal{L}^{\text{temp}} = \sum_{j=1}^{s_l} KL(\mathbf{p}_{t,j}^{\text{temp}} || \mathbf{p}_{s,j}^{\text{temp}}). \quad (3)$$

The instance loss contrasts $\mathbf{s}_j$ with the representations of buffered sequences at temporal position j (red dotted arrows). We obtain the l-dimensional student probability distribution

$$\mathbf{p}_{s,j}^{\text{inst}}(k) = \frac{\exp(\text{sim}(\mathbf{s}_j, \mathbf{q}_j^k)/\tau)}{\sum_{m=1}^{l} \exp(\text{sim}(\mathbf{s}_j, \mathbf{q}_j^m)/\tau)}, \quad (4)$$

and the teacher probability distribution

$$\mathbf{p}_{t,j}^{\text{inst}}(k) = \frac{\exp(\text{sim}(\mathbf{t}_j, \mathbf{q}_j^k)/\tau)}{\sum_{m=1}^{l} \exp(\text{sim}(\mathbf{t}_j, \mathbf{q}_j^m)/\tau)}, \quad (5)$$

where $\mathbf{q}^k$ denotes the kth anchor sequence in the memory buffer. The instance loss is then obtained by summing the KL divergences $KL(\mathbf{p}_t^{\text{inst}} || \mathbf{p}_s^{\text{inst}})$ over all temporal positions:

$$\mathcal{L}^{\text{inst}} = \sum_{j=1}^{s_l} KL(\mathbf{p}_{t,j}^{\text{inst}} || \mathbf{p}_{s,j}^{\text{inst}}). \quad (6)$$

The overall self-supervised loss is given by

$$\mathcal{L} = \alpha \cdot \mathcal{L}^{\text{inst}} + (1 - \alpha) \cdot \mathcal{L}^{\text{temp}}, \quad (7)$$

where α is a balancing hyperparameter.

4. Experiments

In this section, we present our experimental results using the pre-trained representations learned by our proposed model.

Table 1. Average metrics on classfication. Results are reported in percentage.

Model	Dataset	Acc. ↑	AUPRC ↑
TS2Vec (Yue et al., 2022)	UCR	**82.0**	**84.0**
Ours	UCR	78.8	79.7
TS2Vec (Yue et al., 2022)	UEA	**71.2**	**70.6**
Ours	UEA	69.6	70.4

Table 2. The task of anomaly detection on KPI. Results are reported in percentage.

Model	Precision ↑	Recall ↑	F1 ↑
TS2Vec	**92.9**	53.3	67.7
Ours	91.8	**55.2**	**68.9**

4.1. Tasks, datasets, and evaluation metrics

We evaluate our model on three different downstream tasks:

Classification: For this task, we use the 125 UCR[1] and 30 UEA[2] benchmarks, which consist of many small datasets of univariate time series. To evaluate the performance of our model, we use accuracy.

Anomaly detection: For this task, we use the KPI dataset (Ren et al., 2019), a competition dataset released by the AIOPS Challenge. It consists of multiple KPI (key performance indicator) curves from 58 companies with very long sequences varying from 4,000 to 75,000 for the anomaly detection task. We use precision, recall, and F1 score to evaluate the performance.

Forecasting: For this task, we use the 3ETT[3] (Zhou et al., 2021) and Electricity[4] (Dua & Graff, 2017) datasets. We perform both univariate and multivariate forecasting and evaluate the performance by calculating the mean-squared-error (MSE) and mean-absolute-error (MAE).

4.2. Hyperparameter settings

The size of the queue is 128 in all experiments, the temperature τ is 0.07, and the temporal and instance losses have the same weight in optimization, i.e., $\alpha = 0.5$. For all hyperparameters in common with TS2Vec, we follow the hyperparameter settings reported in Yue et al. (2022). The same pre-trained representations are used for all three downstream tasks.

[1] https://www.cs.ucr.edu/~eamonn/time_series_data_2018

[2] http://www.timeseriesclassification.com/

[3] https://github.com/zhouhaoyi/ETTDataset

[4] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

Table 3. Time series forecasting on 3ETT and Electricity datasets. The metrics are averaged over 5 different horizons per dataset. Detailed results are included in Table 7 and Table 8 in the supplementary.

Univariate time series forecasting

Model	Dataset	avg. MSE ↓	avg. MAE ↓
TS2Vec	ETTh1	0.110	0.252
Ours	ETTh1	**0.105**	**0.247**
TS2Vec	ETTh2	**0.170**	**0.321**
Ours	ETTh2	0.173	0.324
TS2Vec	ETTm1	0.069	0.186
Ours	ETTm1	**0.062**	**0.178**
TS2Vec	Electricity	0.486	0.425
Ours	Electricity	**0.485**	**0.419**

Multivariate time series forecasting

Model	Dataset	avg. MSE ↓	avg. MAE ↓
TS2Vec	ETTh1	0.788	0.646
Ours	ETTh1	**0.752**	**0.639**
TS2Vec	ETTh2	**1.567**	**0.937**
Ours	ETTh2	2.020	1.098
TS2Vec	ETTm1	0.628	0.552
Ours	ETTm1	**0.565**	**0.532**
TS2Vec	Electricity	0.330	0.405
Ours	Electricity	**0.317**	**0.397**

4.3. Results

Table 1 shows the overall performance of our full model on the UCR and UEA datasets for classification. TS2Vec outperforms our model by a small margin. Interestingly, our results show that adding the hierarchical contrast (as recommended in Yue et al. (2022)) does not improve our model for classification (details in Section 4.4).

Table 2 shows experimental results on the KPI dataset for anomaly detection. On this task, our model achieves higher recall and F1 scores, while TS2Vec performs better on precision.

Table 3 shows experimental results on the 3ETT and Electricity datasets for univariate and multivariate time series forecasting. On both univariate and multivariate time series forecasting, our model obtains better MSE and MAE on three of the four datasets (ETTh1, ETTm1, and Electricity).

4.4. Ablations

We next analyze the impact of similarity distillation in temporal and instance dimensions. Table 4 presents an ablation study of our approach on all downstream tasks, showing the performance of the temporal loss, instance loss, and the full model with both losses. We also provide ablations on the hierarchical loss as proposed in TS2Vec. On the classification and anomaly detection tasks, we find that the temporal and instance losses are both important, with the full model obtaining the best results. On the forecasting task, the combined loss does not improve results. Hierarchical contrast boosts the performance on the anomaly detection and forecasting tasks by a small margin.

We include additional sensitivity experiments on the queue size and temperature hyperparameters in the supplementary.

Table 4. Ablations on loss functions on the downstream tasks of classification, anomaly detection, and forecasting. Note that the same self-supervised representations are used for all three downstream tasks.

Task: Classification

Model	Dataset	Acc. ↑	AUPRC ↑
Instance only	UCR	77.7	78.4
Temporal only	UCR	78.4	79.4
Full model	UCR	**78.8**	**79.7**
Full + Hierarchical	UCR	77.8	79.0
Instance only	UEA	66.8	68.6
Temporal only	UEA	**70.0**	69.9
Full model	UEA	69.6	**70.4**
Full + Hierarchical	UEA	67.7	69.8

Task: Anomaly Detection

Model	Precision ↑	Recall ↑	F1 ↑
Instance only	91.8	48.0	63.0
Temporal only	91.8	54.5	68.4
Full model	91.8	55.2	68.9
Full + Hierarchical	**92.7**	**55.3**	**69.3**

Task: Forecasting

Model	Type	MSE ↓	MAE ↓
Instance only	uni	0.211	0.301
Temporal only	uni	0.206	0.**292**
Full model	uni	0.206	**0.292**
Full + Hierarchical	uni	**0.205**	0.293
Instance only	multi	0.846	0.646
Temporal only	multi	0.903	0.665
Full model	multi	0.914	0.667
Inst. + Hierarchical	multi	**0.837**	**0.639**

5. Conclusion

We introduced a similarity distillation based self-supervised method for pre-training universal time series representations. In future work, we plan to broaden our exploration into different self-supervised augmentation options.

References

Chen, X. and He, K. Exploring simple siamese representation learning. In *Conference on Computer Vision and Pattern Recognition*, 2021.

Dua, D. and Graff, C. Uci machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

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 *International Joint Conference on Artificial Intelligence*, pp. 2352–2359, 2021.

Franceschi, J., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series. In *Advances in Neural Information Processing Systems*, pp. 4652–4663, 2019.

Malhotra, P., TV, V., Vig, L., Agarwal, P., and Shroff, G. TimeNet: Pre-trained deep recurrent neural network for time series classification. *arXiv preprint arXive:1706.08838*, 2017.

Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., and Zhang, Q. Time-series anomaly detection service at microsoft. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2019. doi: 10.1145/3292500.3330680.

Tejankar, A., Koohpayegani, S. A., Pillai, V., Favaro, P., and Pirsiavash, H. ISD: Self-supervised learning by iterative similarity distillation. In *International Conference on Computer Vision*, pp. 9609–9618, October 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.

Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., and Xu, B. TS2Vec: Towards universal representation of time series. *AAAI Conference on Artificial Intelligence*, 2022.

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., and Eickhoff, C. A transformer-based framework for multivariate time series representation learning. In *International Conference on Knowledge Discovery & Data Mining*, pp. 2114–2124. ACM, 2021.

Zheng, M., You, S., Wang, F., Qian, C., Zhang, C., Wang, X., and Xu, C. ReSSL: Relational self-supervised learning with weak augmentation. In *Advances in Neural Information Processing Systems*, December 2021.

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 *AAAI Conference on Artificial Intelligence*, 2021.

Supplementary Material

A. Sensitivity Analyses

We performed sensitivity analyses on temperature and queue size to evaluate their impact on different tasks. Table 5 shows a set of different temperatures for both student (s_τ) and teacher (t_τ) networks on the anomaly detection task. Using different temperatures for student and teacher networks can improve the final performance for anomaly detection. However, using the best temperature on anomaly detection does not improve the results on other tasks. Finding a universal set of best hyperparameters for all tasks remains a challenge.

Table 5. Impact of temperature. Bottom rows for each task show the original setting used in the model. The best temperatures from the anomaly detection task are used on other downstream tasks to evaluate performance.

Task: Anomaly Detection

s_τ	t_τ	Precision ↑	Recall ↑	F1 ↑
0.1	0.01	92.1	53.6	67.7
0.4	0.04	89.7	57.2	69.9
0.7	**0.07**	90.7	**59.1**	**71.6**
0.01	0.01	92.2	55.1	68.9
0.05	0.05	91.9	54.7	68.6
0.1	0.1	91.7	56.5	69.9
0.2	0.2	**92.3**	55.7	69.5
0.07	0.07	91.8	55.2	68.9

Task: Classification

s_τ	t_τ	Dataset	Acc. ↑	AUPRC ↑
0.7	0.07	UCR	77.5	78.2
0.7	0.07	UEA	68.3	69.8
0.07	0.07	UCR	**78.8**	**79.7**
0.07	0.07	UEA	**69.6**	**70.4**

Task: Forecasting

s_τ	t_τ	Type	avg. MSE ↓	avg. MAE ↓
0.7	0.07	uni	0.216	0.299
0.7	0.07	multi	0.952	0.683
0.07	0.07	uni	**0.206**	**0.292**
0.07	0.07	multi	**0.914**	**0.667**

We analyzed the impact of queue size on a subset of datasets. Table 6 shows that increasing queue size can help with classification and anomaly detection while there seems to be a sweet spot. On the other hand, it seems that queue size does not have a significant impact on forecasting. We obtain modestly better results if we use a smaller queue size for forecasting.

Table 6. Impact of queue size on different tasks.

Task: Classification on UCR.

Queue	Acc. ↑	AUPRC ↑
64	78.6	79.9
128	78.8	79.7
256	**79.1**	**80.1**
512	78.9	79.6

Task: Anomaly Detection.

Queue	Precision ↑	Recall ↑	F1 ↑
64	91.9	54.8	68.6
128	91.8	55.2	68.9
256	91.9	**56.0**	**69.6**
512	91.7	55.3	69.0

Task: Univariate Forecasting on ETTh1.

Queue	H	MSE ↓	MAE ↓
64	24	0.045	0.164
128	24	*0.045*	*0.163*
256	24	0.046	0.165
64	48	*0.068*	*0.201*
128	48	0.068	0.202
256	48	0.069	0.203
64	168	*0.115*	*0.260*
128	168	0.116	0.262
256	168	0.117	0.263
64	336	*0.131*	*0.280*
128	336	0.132	0.282
256	336	0.132	0.282
64	720	*0.161*	*0.323*
128	720	0.163	0.325
256	720	0.163	0.325

B. Detailed forecasting results on each horizon

Table 7 and Table 8 show detailed forecasting results on different horizons.

Table 7. Univariate forecasting per horizon and dataset.

dataset	H	Ours		TS2Vec (Yue et al., 2022)	
		MSE ↓	MAE ↓	MSE ↓	MAE ↓
ETTm1	24	0.016	0.094	**0.015**	**0.092**
ETTm1	48	0.027	**0.123**	0.027	0.126
ETTm1	96	0.046	164	**0.044**	**0.161**
ETTm1	288	**0.092**	**0.233**	0.103	0.246
ETTm1	672	**0.129**	**0.275**	0.156	0.307
ETTh1	24	0.045	0.163	**0.039**	**0.152**
ETTh1	48	0.068	0.202	**0.062**	**0.191**
ETTh1	168	**0.116**	**0.262**	0.134	0.282
ETTh1	336	**0.132**	**0.282**	0.154	0.310
ETTh1	720	0.163	**0.325**	0.163	0.327
ETTh2	24	0.093	0.231	**0.090**	**0.229**
ETTh2	48	0.128	0.275	**0.124**	**0.273**
ETTh2	168	**0.205**	**0.359**	0.208	0.360
ETTh2	336	0.213	0.369	0.213	0.369
ETTh2	720	0.226	0.385	**0.214**	**0.374**
Electricity	24	**0.259**	**0.285**	0.260	0.288
Electricity	48	**0.310**	**0.316**	0.319	0.324
Electricity	168	**0.424**	**0.385**	0.427	0.394
Electricity	336	0.569	**0.465**	**0.565**	0.474
Electricity	720	0.863	0.646	**0.861**	**0.643**

Table 8. Multivariate forecasting per horizon and dataset.

dataset	H	Ours		TS2Vec (Yue et al., 2022)	
		MSE ↓	MAE ↓	MSE ↓	MAE ↓
ETTm1	24	**0.393**	**0.418**	0.443	0.436
ETTm1	48	**0.509**	**0.492**	0.582	0.515
ETTm1	96	**0.551**	**0.526**	0.622	0.549
ETTm1	288	**0.635**	**0.583**	0.709	0.609
ETTm1	672	**0.734**	**0.641**	0.786	0.655
ETTh1	24	**0.533**	**0.516**	0.599	0.534
ETTh1	48	**0.571**	**0.540**	0.629	0.555
ETTh1	168	**0.721**	**0.629**	0.755	0.636
ETTh1	336	**0.886**	**0.713**	0.907	0.717
ETTh1	720	1.049	0.795	**1.048**	**0.790**
ETTh2	24	0.544	0.564	**0.398**	0.461
ETTh2	48	0.782	0.691	**0.580**	**0.573**
ETTh2	168	2.551	1.258	**1.901**	**1.065**
ETTh2	336	3.006	1.431	**2.304**	**1.215**
ETTh2	720	3.217	1.548	**2.650**	**1.373**
Electricity	24	**0.273**	**0.365**	0.287	0.374
Electricity	48	**0.294**	**0.380**	0.307	0.388
Electricity	168	**0.320**	**0.400**	0.332	0.407
Electricity	336	**0.337**	**0.413**	0.349	0.420
Electricity	720	**0.362**	**0.430**	0.375	0.438