UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2022

Reproducibility Study of "TS2Vec: Towards Universal Representation of Time Series"

University of Michigan

Edited by
(Editor)

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
03 February 2023

Published
–

DOI
–

Reproducibility Summary

Scope of reproducibility – We evaluate the reproducibility of the paper *TS2Vec: Towards Universal Representation of Time Series* by Yue et. al., in order to verify their main claims, which state (i) TS2Vec outperforms SOTAs on benchmark time series tasks, specifically, classification and forecasting, and (ii) their proposed framework, TS2Vec, can learn contextual representations of time series data for arbitrary sub-series at various semantic levels, which is useful for accomplishing the aforementioned benchmark tasks.

Methodology – Using the published TS2Vec codebase [1] and referenced time series datasets, we reproduced most of the forecasting (univariate and multivariate) and a portion of the classification results presented in the paper. We then investigated the performance of TS2Vec on forecasting a new time series measuring groundwater level in a rain garden. We conducted a hyperparameter search and modified the source code to make the forecasting results visualizable. Our reproducibility study comes at a total computational cost of 3.163 GPU hours.

Results – We confirmed claim (i), by reproducing the reported accuracy of univariate forecasting, multivariate forecasting, and classification to within -2.39%, -3.77%, and -0.56% of the reported values, supporting the claim that TS2Vec outperforms existing SOTAs. Our investigation into claim (ii), centered around showing TS2Vec's transferability for forecasting the new dataset, has yielded poor results. We have found that in spite of reporting good metrics, TS2Vec may have previously unidentified limitations.

What was easy – The original code implementation was publicly accessible and well structured. Replicating the experiments reported in the paper was straightforward.

What was difficult – Minimal sample code is provided to support applying the algorithm to other datasets. Successfully training a model and then visualizing forecasting on a new dataset was nontrivial. Script arguments were built out to enable modification of some hyperparameters, but not all. We had to edit the source code to complete our investigation and visualize our results.

Communication with original authors – We requested and received the appendix from the corresponding author. We also requested the code for reproducing the plots in the manuscript but did not get a response.

Correspondence should be addressed to ()
The authors have declared that no competing interests exists.
Code is available at https://github.com/rescience-c/template.

1 Introduction

Time series datasets are critically important in various industries, including the financial and energy sectors. Performing classification, forecasting, and anomaly detection on time series data are fundamental but challenging tasks. In comparison to other types of data, time series data is particularly susceptible to quality issues such as noise and data gaps. One strategy for working with this data is to learn a "representation" of the time series, which can then be processed on in lieu of the actual data to achieve better results. Past work has focused on learning instance-level (e.g., for the entire length of the input time series) representations of the data [2, 3, 4] but there are limitations with these methods, particularly when applied to tasks such as forecasting and anomaly detection. Most existing methods rely on assumptions like transformation-invariance and cropping-invariance, which are not always applicable.

To address these limitations, Yue et al. [5] propose a contrastive learning framework called TS2Vec, which enables the representation learning of time series at all semantic levels. TS2Vec uses hierarchical contrasting on both the spatial and temporal dimensions to capture multi-scale contextual information. It then utilizes contextual consistency for positive pair selection. The representation of an arbitrary sub-sequence in the time series can then be obtained with a simple aggregation (i.e., max pooling) over the representations of the corresponding timestamps.

We have reproduced a significant fraction of the results reported in the paper and have performed additional experiments to verify their claims. This work makes the following contributions:

- We reproduce the forecasting and classification tasks on the same datasets as the manuscript using TS2Vec and compare them to the reported values in the paper.
- We extend upon the work by evaluating how TS2Vec forecasting performs on a new time series dataset measuring groundwater depth in a rain garden.

2 Scope of Reproducibility

TS2Vec is presented as a universal framework for learning representations of time series at an arbitrary semantic level. The authors claim that by performing contrastive learning in a hierarchical way, the contextual representation is more robust for each timestamp, which improves the accuracy of many time series tasks. Yue et al. [5] claims it is the first work that provides a flexible and universal representation method for all kinds of tasks in the time series domain, including but not limited to time series classification, forecasting, and anomaly detection. In this reproducibility study, our main goal is to verify the following claims of the original paper:

- **Performance:** TS2Vec outperforms existing SOTAs on benchmark time series tasks, including classification and forecasting.
- **Transferability:** TS2Vec learns a *useful* model and contextual representation of arbitrary sub-series at various semantic levels for datasets beyond those reported on in the original paper.

The following is the layout of the remainder of this work. Section 3 details our approach to reproduce the original paper's performance metrics for classification and forecasting. It also details our methodology for evaluating the practicality of TS2Vec by examining the performance of the algorithm on a new dataset, the Open-Storm Hydrologic dataset for forecasting [6]. Section 4 presents the replicated results and the results obtained

from the new dataset. These are both compared to the reported findings of the original paper. Section 5 concludes this work with a discussion on TS2Vec.

3 Methodology

We used the TS2Vec code, which is freely available on Github [1] for our reproducibility experiments. We tested the algorithm in two ways: (1) we used the original code and datasets to reproduce the paper's reported results and (2) we critically examined the performance of TS2Vec on a new dataset.

3.1 Model descriptions

Given a set of time series $X = x_1, x_2, ..., x_N$ of N instances, the goal is to learn a nonlinear embedding function f_θ that maps each x_i to its representation r_i that best describes itself. The input time series x_i has dimension $T \times F$ (T is the sequence length, F is the feature dimension). The representation $r_i = r_{i,1}, r_{i,2}, ..., r_{i,T}$ contains representation vectors $r_{i,t} \in \mathbb{R}^{\mathbb{K}}$ for each timestamp t.

The architecture of TS2Vec is shown in Figure 1, taken from the original paper [5]. First, two overlapping sub-series are randomly sampled from an input time series x_i, this is called random cropping. Random cropping creates random subsets, also called contexts, of the time series which enables the comparison of overlapping segments for the training phase. For any time series input x_i, TS2Vec randomly samples two overlapping time segments $[a_1, b_1], [a_2, b_2]$ such that $0 < a_1 \leq a_2 \leq b_1 \leq b_2 \leq T$. The contextual representations on the overlapped segment $[a_2, b_1]$ should be consistent for two context reviews. Overall, random cropping increases the amount of training data, which helps avoid representation collapse and removes positional information that is implicitly encoded into the representation.

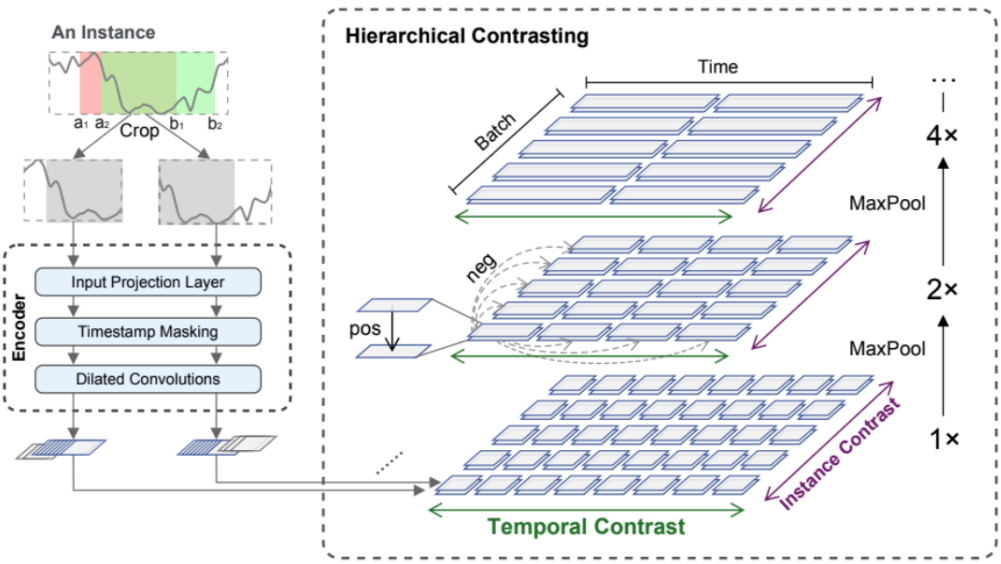

Figure 1. TS2Vec architecture showing a univariate time series as the input example, the framework also supports multivariate input. Each parallelogram denotes the representation vector on a timestamp of an instance. (Figure 1 from the original paper [5])

These random subsets are passed to the encoder. The encoder f_θ consists of three components: input projection layer, a timestamp masking module, and a dilated CNN mod-

ule. For each input x_i, the input projection layer is a fully connected layer that maps the observation $x_{i,t}$ at timestamp t to a higher-dimensional latent vector $z_{i,t}$. This allows for a better generalization and enables comparison of various time series. Then, the timestamp masking module masks the latent vector $z_i = \{z_{i,t}\}$ along the time axis with a binary mask $m \in \{0,1\}^T$, the elements of which are independently sampled from a Bernoulli distribution with $p = 0.5$. Timestamp masking forces the model to determine which parts of the time series subsets are the same based on features instead of the time feature. The dilated CNN module with ten residual blocks is then applied to extract the contextual representation at each timestamp. Each block contains two 1-D convolution layers with a dilation parameter (2^ℓ for the ℓ-th block). The dilated convolutions enable a large receptive field for different domains.

The encoder is optimized jointly with temporal and instance-wise contrastive loss, with the total loss being summed over multiple scales in a hierarchical framework. Hierarchical contrasting is used to change the resolution of two randomly sampled overlapping sub-series used for positive pair selection. To do this, max pooling is applied on the learned representations along the time axis and Equation 3 is computed recursively. The loss functions are applied at all granularity levels in the hierarchical contrasting model. Specifically, temporal contrastive loss compares signal dynamics in two subsets (contexts) of the same time series. The representations at the same timestamp are marked as positive pairs while any pair with different timestamps are marked as negative. $r_{i,t}$ and $r'_{i,t}$ are the representations for the same timestamp t but from two augmentations of x_i. The temporal constrastive loss for the i-th time series at timestamp t is

$$\ell_{temp}^{(i,t)} = -log\frac{exp(r_{i,t} \cdot r_{i,t}^{'})}{\sum_{t' \epsilon \Omega}(exp(r_{i,t} \cdot r_{i,t'}^{'}) + 1_{[t \neq t']} exp(r_{i,t} \cdot r_{i,t'}))}, \tag{1}$$

where Ω is the set of timestamps within the overlap of the two sub-series of the same instance, and $\mathbf{1}$ is the indicator function. The other loss function, instance-wise contrastive loss, is used to compare instance-specific characteristics across multiple time series. The equation for the instance-wise loss of time series i at timestamp t is

$$\ell_{inst}^{(i,t)} = -log\frac{exp(r_{i,t} \cdot r_{i,t}^{'})}{\sum_{j=1}^{B}(exp(r_{i,t} \cdot r_{j,t}^{'}) + 1_{[i \neq j]} exp(r_{i,t} \cdot r_{j,t}))}, \tag{2}$$

where B denotes the batch size. Representations of other time series at timestamp t are marked as negative samples. The two losses are complementary to each other. The overall loss is defined as

$$\mathcal{L}_{dual} = \frac{1}{NT}\sum_i \sum_t (\ell_{inst}^{(i,t)} + \ell_{temp}^{(i,t)}). \tag{3}$$

Overall, the TS2Vec algorithm learns a vector representation of a time series dataset by contrasting randomly selected overlapping subsets of the series against each other. This vector representation can then be used to interpret the data for benchmark time series tasks such as forecasting and classification. For the time series classification tasks, the classes are labeled on the entire time series (instance). To obtain the instance level representations, max pooling is done over all timestamps. An SVM classifier with RBF kernel is trained on top of the instance-level representations to make predictions. For the time series forecasting tasks, given the last T_ℓ observations $x_{t-T_\ell+1}, ...x_t$, time series forecasting aims to predict the future H observations $x_{t+1}, ...x_{t+H}$. To predict future observations, we use r_t, the representation of the last timestamp. Specifically, we train a linear ridge regression model with L_2 norm penalty that takes r_t as input to directly predict future values x^{hat}. When x is a univariate time series, x^{hat} has dimension H. When x is a multivariate time series with F features, the dimension of x^{hat} is FH.

3.2 Datasets

We used the same datasets evaluated by Yue et al. [5] to validate the reported model performance metrics. All of the datasets were preprocessed via scripts provided on Github [1]. For the classification task, we validated 64 of the 128 univariate datasets in the UCR archive [7]. The UCR archive is a community driven resource that consists of individual time series datasets of images, sensor measurements, simulations, and more to test classification algorithms. The train/val/test ratios varies per dataset, more information can be found in Dau et. al [7]. For the forecasting tasks, we validated the 3 univariate and multivariate ETT datasets [8] and the univariate Electricity dataset [9]. The ETT datasets are 2-years of power transformer data from two stations at different time intervals. The Electricity dataset contains the electricity consumption data of 321 clients over three years. The train/val/test split was 12/4/4 months for the ETT datasets and 60/20/20 for the Electricity dataset.

A publicly-available hydrologic dataset [6] was used to examine the transferability of the TS2Vec framework and to more closely examine the quality of forecasting. The dataset is a multivariate time series collected from a single site (single instance) measuring groundwater levels and rainfall in a rain garden (10,364 measurements over 3 months). Each variable was normalized using z-score and formatted to match the Electricity dataset used in the original paper. A 60/20/20 percent train/val/test split was used. For some experiments, the groundwater depth dataset was feature engineered to be add more features (see Section 3.3 for details).

3.3 Hyperparameters

The TS2Vec codebase provides arguments to support hyperparameter selection for a variety of variables during the training of the model. We identified 5 hyperparameter arguments that could affect model performance, which included batch size, learning rate (lr), number of representation dimensions (repr-dims), max training length, and epochs. We also identified several other relevant hyperparameters that were not offered as arguments to the training function. These included the weighting of temporal versus instance-wise contrastive loss in the overall loss function (α), the ridge regression coefficient used when fitting the model (also α, by the paper's definition), batch size (distinct from training batch size) as well as the padding and sliding length variables used during forecasting.

Reproduction of paper results — When reproducing the results presented in the paper, we adhered to the argument hyperparameters specified in the `scripts/` folder on Github [1]. Within the TS2Vec codebase, sklearn's `GridSearchCV` function [10] is used to select the best hyperparameters for the estimator used to fit the model to the data. The default estimator for classification is Support Vector Machines (SVM), and for forecasting is Ridge Regression. We modified the source code to print the optimal estimator hyperparameters used in each time series task. See Appendix Sections 6.1 and 6.2 for further details.

Forecasting on a new dataset — After modifying the source code to expose the non-argument hyperparameters, we searched over the hyperparameter space manually, paying less attention to the reported MSE and MAE values and more to the qualitative fit of the forecast to the data. We were looking for a combination of hyperparameters that resulted in a "useful" forecast — one that consistently matched the shape and magnitude of the groundwater data. In total, over 40 models were trained and evaluated during the hyperparameter search. Further details on the hyperparameter search for forecasting on a new dataset can be found in Appendix Table 13.

3.4 Experimental setup and code

To reproduce the results of the original paper, we used the same forecasting and classification functions, datasets, and performance metrics provided in the TS2Vec Github [1] to validate the reported model performance. We created the appropriate python environment in Google Colab and then we used the lines of code provided in the `scripts` folder to run the forecasting and classification tasks. We compared our performance metric values to the TS2Vec and SOTA values obtained by Yue et al. [5]. For forecasting tasks, we use sklearn's root mean squared error (MSE) and mean absolute error (MAE) [10]. For classification tasks, we use sklearn's accuracy score, which is the accuracy of labels predicted to their true label for each dataset [10].

For our experiment beyond the original paper, we ran the forecasting script on a hydrologic dataset chosen because of its relevance to the reproduction study authors' research. Our objective was to investigate the transferability of the TS2Vec framework (claim ii) by observing how well the model forecasts this data, which happened to be much less complex than the original paper's visualized EET dataset. We conducted a hyperparameter search for forecasting the groundwater depth dataset as discussed in Section 3.3. We also investigated the impact of feature engineering, training models on modified versions of the groundwater dataset that included rainfall data, the first derivative of depth, and lookback columns up to 10 timestamps in the past. We visualized the performance of each forecasting experiment for both the best-performing (lowest MSE) slice as well as a random slice of the dataset. Our code can be found at github.com/jacquelynq/MLReproducibilityChallenge22_TS2Vec.

3.5 Computational requirements

The original implementation requires `torch.cuda` enabled, which limits the machines that can run the authors' code to those with CUDA-capable graphics cards. As a result, we used a Google Colab notebook with runtime type set to GPU. With appropriate modifications to the `init` function of `ts2vec.py` (e.g. `device='gpu'`), however, we were able to run the code on our personal computers, albeit with a much slower runtime. Our reproducibility study comes at a total computational cost of 3.163 GPU hours.

We found the provided `requirements.txt` file was insufficient. The `bottleneck` library requirement was capitalized in the file, which broke the `pip` install. In addition to the listed requirements, we also had to install the `statsmodels` and `tables` libraries. We found that `python==3.8` and `scikit_learn==0.24.2` were hard requirements, but more recent versions of other libraries were compatible.

How to get access to the original datasets used in the paper is well documented in the Github `README.md`, however, downloading these datasets comes with a few caveats. The UCR image classification archive is free to use with accreditation, however, the `.zip` file is password protected. The password can be found by reading through the documentation provided on the UCR homepage. The Yahoo dataset can only be obtained on request by filling out a request form. They restrict the use of this dataset to non-commercial applications and ask for a full name and contact information.

4 Results

4.1 Results reproducing original paper

Time series forecasting — To validate the forecasting ability of TS2Vec, we ran their code and calculated the MSE and MAE for all four univariate time series datasets: Electicity,

$ETTh_1$, $ETTh_2$, and $ETTm_1$ (Table 1) as well as for three of the multivariate time series datasets: $ETTh_1$, $ETTh_2$, and $ETTm_1$ (Table 2). The multivariate Electricity time series dataset timed out in Google Colab so we were unable to reproduce these results. Overall, our univariate and multivariate forecasting MSE values were within -2.39% and -3.77% (percent difference) of the values provided by Yue et al. [5], respectively. See Appendix Section 6.2 for the full results. We also provide the MSE and MAE values for the SOTAs evaluated by Yue et. al. [5] in Tables 9, 10, 11, and 12. Overall, our average MSE and MAE values were lower TS2Vec and all SOTAs reported in [5]. We provide our training times for the forecasting tasks in Table 8. In Table 8 we also report the training times for TS2Vec and the SOTA, Informer, for the multivariate EET_{m1} dataset. Our training time on the multivariate EET_{m1} dataset was faster than the values reported for TS2Vec and Informer [5]. However, we cannot adequately compare the training times of the other datasets because Yue et al. did not provide them [5].

	TS2Vec	Our TS2Vec	Informer	LogTrans	N-BEATS	TCN	LSTnet
Avg MSE	0.209	0.204	0.310	0.370	0.399	0.338	1.099
Avg MAE	0.296	0.295	0.390	0.434	0.426	0.413	0.536

Table 1. Average MSE and MAE originally reported for TS2Vec and the SOTAs in [5], as well as our reproduction of TS2Vec for univariate time series forecasting.

	TS2Vec	Our TS2Vec	Informer	LogTrans	N-BEATS	TCN	LSTnet
Avg MSE	0.994	0.957	1.342	1.136	1.470	1.635	2.411
Avg MAE	0.712	0.702	0.859	0.778	0.968	1.045	1.730

Table 2. Average MSE and MAE originally reported for TS2Vec and the SOTAs in [5], as well as our reproduction of TS2Vec for multivariate time series forecasting.

Time series classification — We reproduced 50% of the UCR dataset in order to validate the claim that TS2Vec out preformed SOTAs in classifying time series data. The average accuracy score for these 64 datasets are reported in Table 3. Overall, our results were within -0.56% (percent difference) of TS2Vec's accuracy score reported by Yue et al. [5]. The average accuracy score in classifying 50% of the UCR datasets was highest by TS2Vec, then our reproduction of TS2Vec, followed by the SOTAs reported in [5]. Table 3 also provides the total training time (hours) for TS2Vec and the SOTAs on the full UCR dataset. Note, our reproduced TS2Vec training time is for 50% of the UCR dataset so it is had to compare our trainign time to TS2Vec or the SOTAs. See Appendix Section 6.1 for the accuracy score of each individual UCR dataset by TS2Vec, our reproduction of TS2Vec, and the SOTAs.

Method	TS2Vec	Our TS2Vec	T-Loss	TNC	TS-TCC	TST	DTW
Avg Accuracy	0.882	0.877	0.871	0.839	0.843	0.729	0.794
Training Time (hr)	0.900	0.011	38.0	228.4	1.1	17.1	N/A

Table 3. Average accuracy and total training time (hours) for time series classification on 50% of the UCR datasets by TS2Vec, our reproduction of TS2Vec, and the SOTAs compared in [5]. Note the training time for TS2Vec and the SOTAs are on the full UCR dataset while our training time is for 50% of the UCR dataset.

4.2 Results beyond original paper

We did not see compelling forecasting results for any permutation of hyperparameters or features chosen during the hyperparameter search. Little improvement in MSE or vi-

Dataset	H	MSE (norm)	MAE (norm)
Hydrologic	24	0.272	0.227
	48	0.330	0.269
	96	0.367	0.302
	288	0.599	0.430
	672	0.662	0.462

Table 4. Forecasting performance on new hydrologic dataset. H is the prediction length, where H=24 means 24 points are predicted.

sual fit was seen for any of the generated models, although we did note several instances in which the model broke down (e.g. setting learning rate > 0.1). The best fitting slices generated from this process are shown in Figure 2. Forecast fit was observed to be much worse on other slices. Run in the same Google Colab environment as the reproduction results, the model trained in 1:20.9 minutes. The evaluation script took a further 14 seconds. Results are shown in Table 4.

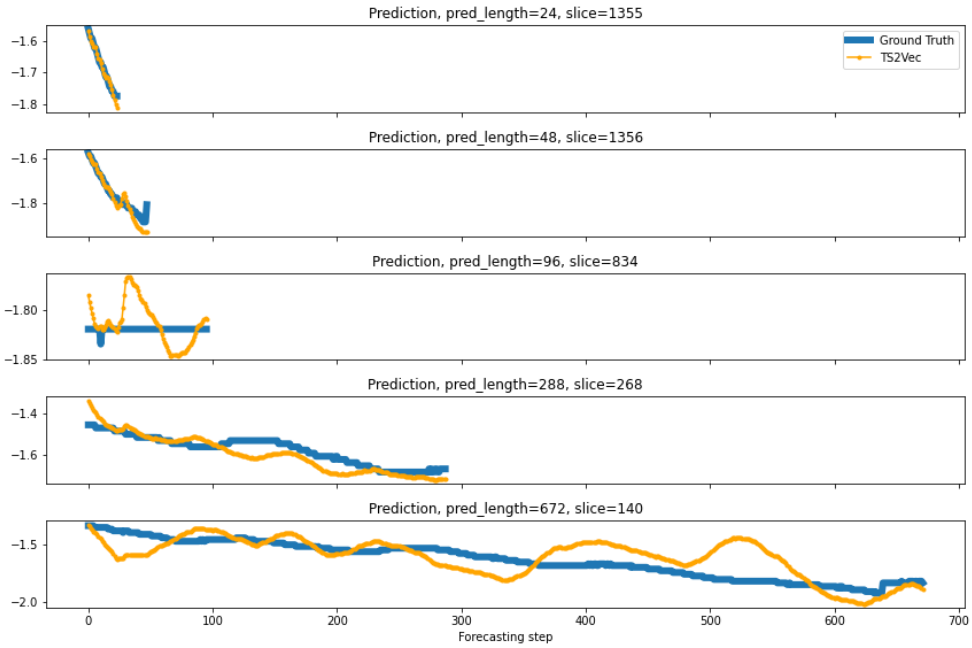

Figure 2. Plot of best prediction slice for each forecasting length for the new hydrologic dataset. The y-axis is an encoded variable.

5 Discussion

Our reproduction of the paper's results validated claim (i), that TS2Vec performs as reported on the datasets used in the paper. Assuming the given performance metrics from the authors' implementation of other SOTAs is correct, we have confirmed that TS2Vec outperforms them in both forecasting and classification. This was what the authors projected due to the TS2Vec model's flexible and universal representation of time series tasks. The TS2Vec's novel contrastive framework which allowed for the increased performance included both hierarchical contrasting and contextual consistency.

Our reproduction was not able to validate claim (ii), that TS2Vec produces a useful representation for arbitrary time series data. Despite conducting an extensive hyperparameter search and experimenting with feature engineering of the groundwater depth dataset, we were not able to train a model that produced an accurate forecast. As a sanity check, we tried running this dataset through a Gated Recurrent Unit (GRU) neural network [11], an older, less complex neural-network based forecasting algorithm. The results of this forecast (see Figure 5) were dramatically better than what we were able to produce using the TS2Vec framework.

It is still unclear why we were not able to produce good forecasting results for our new dataset. Based on a close reading of the paper, we were not able to identify any constraints of the TS2Vec framework that would preclude using it on the groundwater dataset. Further investigation into how well TS2Vec models a wider variety of time series should be conducted, including an examination of how well it models low complexity time series data. It is possible that the added complexity of the composite loss function introduces previously unidentified limitations which reduce TS2Vec's performance in some circumstance.

5.1 What was easy

The TS2Vec code base is very well organized and follows standard coding practices, making it easy to follow and modify. Yue et al. [5] provided the code snippets necessary to reproduce the main results presented in the paper. This greatly simplified the process of confirming their reported performance metrics.

5.2 What was difficult

Google Colab offers limited computational resources to free-tier users. We found that the anomaly detection script was too computationally expensive to run, and as a result were not able to verify the paper's anomaly detection results. For the same reason, we were not able to validate the forecast script for the Electricity multivariate time series dataset. The authors have been thorough in demonstrating the performance of TS2Vec in multiple tasks performed on a wide variety of datasets. Even with code for reproduction provided on Github and reproducing only a fraction of the reported MSE's and MAE's, this step was still quite time-consuming.

While it was straightforward to prepare and run the forecasting script on a new dataset, the code as-is only output performance metrics, not the actual forecast prediction. There was no sample code provided to show how they produced the figures in the paper and they did not demonstrate how to use the forecast script in a practical way. We had to modify the source code to visualize the data (see section 6.3), as well as to modify several hyperparameters that were not given as arguments in the `train.py` script. As described in Section 4.2, we were not able to find a combination of hyperparameters that resulted in an accurate forecast of the groundwater dataset.

5.3 Communication with original authors

We emailed the corresponding author on October 13, 2022 to ask for a version of the paper with the appendix, which we could not find. We received a response with the appropriate link on October 14, 2022. We also asked the corresponding author for the code for reproducing the figures on November 11, 2022. We did not get a response.

References

1. Yuezhihan. "Yuezhihan/ts2vec: A universal time series representation learning framework." In: **GitHub** (2022).
2. S. Tonekaboni, D. Eytan, and A. Goldenberg. "Unsupervised Representation Learning for Time Series with Temporal Neighborhood Coding." In: **International Conference on Artificial Intelligence and Statistics** (2021), pp. 793–802.
3. J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi. "Unsupervised scalable representation learning for multivariate time series." In: **Advances in neural information processing systems** 32 (2019).
4. L. Wu, I. E.-H. Yen, J. Yi, F. Xu, Q. Lei, and M. Witbrock. "Random warping series: A random features method for time-series embedding." In: **International Conference on Artificial Intelligence and Statistics** (2018), pp. 793–802.
5. Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu. "TS2Vec: Towards Universal Representation of Time Series." In: **Proceedings of the AAAI Conference on Artificial Intelligence** 36.8 (2022), pp. 8980–8987.
6. Open-storm. "Open-Storm: Site DET039." In: **Grafana** (2022).
7. H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. "The UCR time series archive." In: **IEEE/CAA Journal of Automatica Sinica** 6 (6 2019).
8. H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting." In: **The Thirty-Fifth AAAI Conference on Artificial Intelligence** (2021).
9. D. Dau and C. Graff. "UCI Machine Learning Repository." In: (2017).
10. F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: **Journal of Machine Learning Research** 12 (2011), pp. 2825–2830.
11. K. Kuguoglu. "Building RNN, LSTM, and GRU for time series using PyTorch." In: **Towards Data Science** (2021).

6 Appendix

6.1 Reproduction of results: Classification

Our optimal hyperparameters and training time for classifying the 64 UCR datasets are provided in Table 5. Our training time on these datasets were very fast, on average 12.7 s, whereas the paper's average for all 128 UCR datasets was 0.9 hr. The discrepancy may be due to the remaining 64 datasets that may be larger and take longer to train. Our accuracy matched the accuracy reported by Yue et al. [5] for 64 UCR datasets (see Table 6. The average accuracy from the paper and our results were 0.877 and 0.882, respectively. Overall, the paper's reported accuracy for TS2Vec outperformed our reproduction of TS2Vec and the 5 SOTAs (see Table 6.

6.2 Reproduction of results: Forecasting

The optimal $alpha$ values for the Ridge Regression estimator for time series forecasting tasks are shown in Table 7. Note, the multivariate Electricity dataset timed out, therefore, we could not report its optimal values. The total training time (s) for the time series forecasting tasks are shown in Table 8. Yue et al. only provided the training time for TS2Vec and Informer on the multivariate $ETTm_1$ dataset. Our training time was faster than both TS2Vec and Informer.

Our MSE and MAE results for each H, the number of points to forecast, within each univariate dataset matched the results reported in Yue et al. [5] (see Table 9 and 10). The overall average MSE on all four univariate datasets was 0.209 (paper) and 0.204 (ours). The overall average MAE on all four univariate datasets was 0.296 (paper) and 0.295 (ours). Overall, either TS2Vec or our reproduced runs of TS2Vec had the best MSE and MAE as compared to the other SOTAs (see bold red numbers in the tables). However, there were a few individual datasets where other SOTAs had better MSE or MAE values (see bold red numbers in the tables). Forecasts for longer prediction lengths seemed to do worse (have more variance) than the shorter prediction lengths, with aligns with

how MSE and MAE grew with prediction length.

Our MSE and MAE results for each H within each multivariate dataset matched the results reported in Yue et al. [5] (see Table 11 and 12). The overall average MSE on the three multivariate datasets was 0.994 (paper) and 0.957 (ours). The overall average MAE on the three multivariate datasets was 0.712 (paper) and 0.702 (ours). Overall, either TS2Vec or our reproduced runs of TS2Vec had the best MSE and MAE as compared to the other SOTAs (see bold red numbers in the tables). However, there were a few individual datasets where other SOTAs had better MSE or MAE values (see bold red numbers in the tables). The multivariate electricity dataset would not run within our Google Collab environment. Forecasts for longer prediction lengths seemed to do worse (have more variance) than the shorter prediction lengths, with aligns with how MSE and MAE grew with prediction length.

6.3 Original result: Forecasting

See Figure 3 — The original result of the $ETTh_2$ forecast for 336 steps at slice 756 was able to be recreated. This ensured that we were using the right outputs from the model to plot our forecasts.

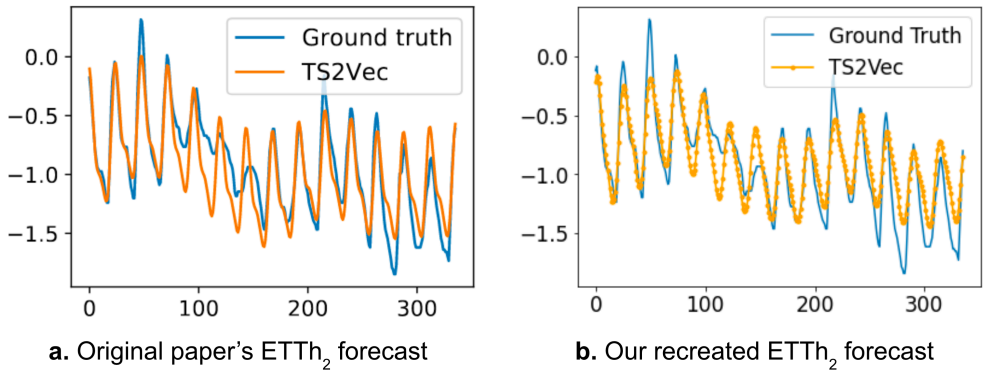

Figure 3. Recreation of EETh2 forecast for 336 forecasted steps. The y-axis is an encoded variable. (Figure **a** from original paper [5])

See Figure 4 — The input dimensions to the CNN model for our multivariate time series data was 10 to account for the encoding of the timestamp as features and the rainfall. See table 13 — Various hyperparameters were adjusted to see the impacts on the groundwater dataset forecast. No combination of hyperparameter tuning was found to produce a reliable forecast for the groundwater forecast.

Hyperparameter search — Table 13 provides details on the hyperparameter search conducted to train a better performing model for forecasting groundwater depth. Each hyperparameter was searched over in the order presented in the table. The only hyperparameters that were judged to improve the fit of the model were `alpha` (ridge-regression), which we wrote code to optimize for each run, `repr-dims` (set to 100, as opposed to default 320), and `epochs` (increased to 400 from default 200). The improvement that these updated hyperparameters brought to the forecasting capability model was minor, however.

Sanity Check — We ran our groundwater depth data through a Gated Recurrent Unit (GRU) neural network with features including the first derivative of depth and ten lookback columns. This much simpler neural network showed great forecasting results.

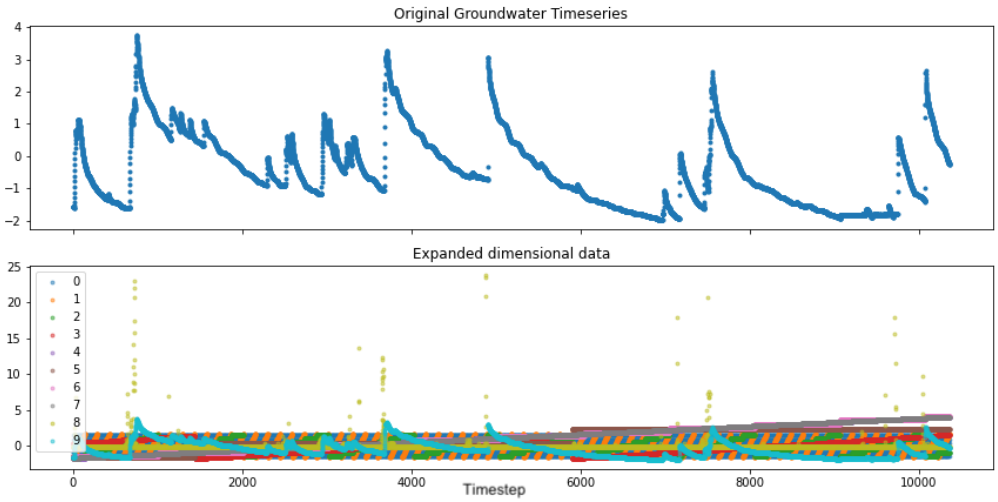

Figure 4. Visualization of original time series and 10-dimensional encoded time series that is input to the CNN. The first eight of the dimensions represent the datetime information, the ninth the rainfall information, and the tenth the groundwater information.The y-axes are encoded variables.

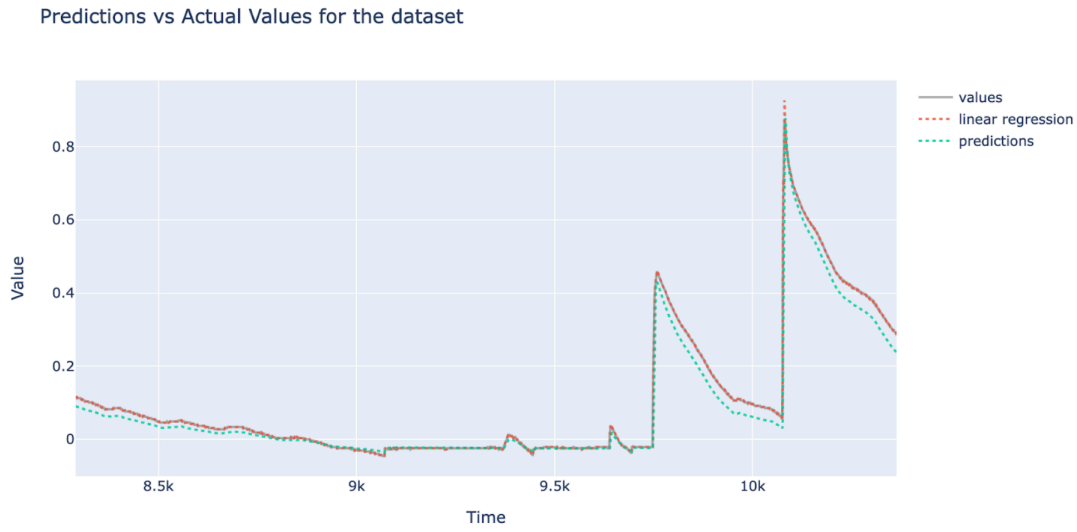

Figure 5. Sanity Check: Forecasting performance of a Gated Recurrent Unit (GRU) NN on the groundwater depth dataset.

Dataset	Optimal C	Our Training Time (s)
Chinatown	∞	9.016
SonyAIBORobotSurface1	∞	9.845
ItalyPowerDemand	10	8.766
MoteStrain	∞	9.908
SonyAIBORobotSurface2	∞	9.529
TwoLeadECG	∞	9.797
SmoothSubspace	10	8.594
ECGFiveDays	∞	10.496
Fungi	∞	10.797
CBF	∞	10.053
BME	∞	13.462
UMD	∞	10.633
DiatomSizeReduction	∞	12.320
DodgerLoopWeekend	∞	14.082
DodgerLoopGame	∞	11.948
GunPoint	100	10.405
Coffee	∞	11.405
FaceFour	∞	12.310
FreezerSmallTrain	∞	11.923
ArrowHead	∞	11.207
ECG200	100	9.853
Symbols	∞	13.079
ShapeletSim	∞	14.286
InsectEPGSmallTrain	∞	15.244
BeetleFly	∞	16.528
BirdChicken	∞	14.264
ToeSegmentation1	∞	11.830
ToeSegmentation2	∞	12.018
Wine	∞	11.078
Beef	∞	16.224
Plane	1	10.209
OliveOil	∞	15.106
SyntheticControl	1	9.293
PickupGestureWiimoteZ	10	12.437
ShakeGestureWiimoteZ	10	12.689
GunPointMaleVersusFemale	10	10.244
GunPointAgeSpan	100	13.434
GunPointOldVersusYoung	0.1	10.092
Lightning7	100	11.635
DodgerLoopDay	10	11.658
PowerCons	10	10.105
FacesUCR	10	9.915
Meat	1000	13.537
Trace	10	11.205
MelbournePedestrian	1000	8.632
MiddlePhalanxTW	100	9.559
DistalPhalanxOutlineAgeGroup	10	9.751
MiddlePhalanxOutlineAgeGroup	10	9.520
ProximalPhalanxTW	1000	9.590
ProximalPhalanxOutlineAgeGroup	10	9.515
DistalPhalanxTW	100	9.610
Herring	10	14.240
Car	10	14.724
InsectEPGRegularTrain	0.1	17.307
MedicalImages	10	9.916
Lightning2	100	15.912
FreezerRegularTrain	10	11.536
Ham	10	12.978
MiddlePhalanxOutlineCorrect	1000	9.665
DistalPhalanxOutlineCorrect	1000	9.535
ProximalPhalanxOutlineCorrect	1000	9.642
Mallat	100	21.742
InsectWingbeatSound	10	10.813
Rock	∞	76.337

Table 5. The training time and optimal C hyperparameter for the SVM classifier for each classification dataset.

Dataset	TS2Vec	Our TS2Vec	T-Loss	TNC	TS-TCC	TST	DTW
Chinatown	0.959	0.971	0.951	0.977	**0.983**	0.937	0.957
SonyAIBORobotSurface1	**0.903**	0.887	0.902	0.804	0.899	0.724	0.725
ItalyPowerDemand	0.925	**0.958**	0.944	0.928	0.955	0.845	0.950
MoteStrain	**0.861**	0.847	0.851	0.825	0.843	0.768	0.835
SonyAIBORobotSurface2	0.871	0.875	**0.889**	0.834	0.907	0.745	0.831
TwoLeadECG	0.986	0.965	**0.999**	0.993	0.976	0.871	0.905
SmoothSubspace	**0.980**	**0.980**	0.960	0.913	0.953	0.827	0.827
ECGFiveDays	**1.000**	**1.000**	**1.000**	0.999	0.878	0.763	0.768
Fungi	0.957	0.914	**1.000**	0.572	0.753	0.366	0.839
CBF	**1.000**	**1.000**	0.983	0.983	0.998	0.898	0.997
BME	**0.993**	0.973	**0.993**	0.973	0.933	0.760	0.900
UMD	**1.000**	**1.000**	0.993	0.993	0.986	0.910	0.993
DiatomSizeReduction	0.984	0.984	0.984	**0.993**	0.977	0.961	0.967
DodgerLoopWeekend	**0.964**	0.957	N/A	N/A	N/A	0.732	0.949
DodgerLoopGame	0.841	0.812	N/A	N/A	N/A	0.696	**0.877**
GunPoint	**0.980**	**0.980**	**0.980**	0.967	0.993	0.827	0.907
Coffee	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.821	**1.000**
FaceFour	**0.932**	0.875	0.920	0.659	0.773	0.511	0.830
FreezerSmallTrain	0.870	0.906	0.933	**0.982**	0.979	0.920	0.753
ArrowHead	**0.857**	0.851	0.766	0.703	0.737	0.771	0.703
ECG200	0.920	0.880	**0.940**	0.830	0.880	0.830	0.770
Symbols	**0.976**	0.972	0.963	0.885	0.916	0.786	0.950
ShapeletSim	**1.000**	0.989	0.672	0.589	0.683	0.489	0.650
InsectEPGSmallTrain	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.735
BeetleFly	0.900	0.850	0.800	0.850	0.800	**1.000**	0.700
BirdChicken	0.800	0.800	**0.850**	0.750	0.650	0.650	0.750
ToeSegmentation1	0.917	**0.934**	0.939	0.864	0.930	0.807	0.772
ToeSegmentation2	0.892	**0.923**	0.900	0.831	0.877	0.615	0.838
Wine	**0.870**	0.852	0.815	0.759	0.778	0.500	0.574
Beef	**0.767**	0.733	0.667	0.733	0.600	0.500	0.633
Plane	**1.000**	0.990	0.990	**1.000**	**1.000**	0.9333	**1.000**
OliveOil	**0.900**	**0.900**	0.867	0.833	0.800	0.800	0.833
SyntheticControl	0.997	0.997	0.987	**1.000**	0.990	0.490	0.993
PickupGestureWiimoteZ	0.820	**0.840**	0.740	0.620	0.600	0.240	0.660
ShakeGestureWiimoteZ	**0.940**	**0.940**	0.920	0.820	0.860	0.760	0.860
GunPointMaleVersusFemale	**1.000**	**1.000**	0.997	0.994	0.997	**1.000**	0.997
GunPointAgeSpan	0.987	0.991	**0.994**	0.984	**0.994**	0.991	0.918
GunPointOldVersusYoung	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.838
Lightning7	**0.863**	0.849	0.795	0.767	0.685	0.411	0.726
DodgerLoopDay	**0.562**	0.525	N/A	N/A	N/A	0.200	0.500
PowerCons	**0.961**	**0.961**	0.900	0.933	**0.961**	0.911	0.878
FacesUCR	0.924	**0.929**	0.884	0.789	0.863	0.543	0.905
Meat	**0.950**	**0.950**	**0.950**	0.917	0.883	0.900	0.933
Trace	**1.000**	**1.000**	0.990	**1.000**	**1.000**	**1.000**	**1.000**
MelbournePedestrian	**0.959**	0.949	0.944	0.942	0.949	0.741	0.791
MiddlePhalanxTW	0.584	0.591	0.591	0.571	**0.610**	0.506	0.506
DistalPhalanxOutlineAgeGroup	0.727	0.719	0.727	0.741	0.755	0.741	**0.770**
MiddlePhalanxOutlineAgeGroup	0.636	0.636	**0.656**	0.643	0.630	0.617	0.500
ProximalPhalanxTW	**0.824**	0.795	0.771	0.810	0.800	0.780	0.761
ProximalPhalanxOutlineAgeGroup	0.834	**0.929**	0.844	0.854	0.839	0.854	0.805
DistalPhalanxTW	**0.698**	0.691	0.676	0.669	0.676	0.568	0.590
Herring	**0.641**	0.563	0.594	0.594	0.594	0.594	0.531
Car	**0.833**	0.717	**0.833**	0.683	0.583	0.550	0.733
InsectEPGRegularTrain	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.872
MedicalImages	0.789	**0.800**	0.750	0.754	0.747	0.632	0.737
Lightning2	**0.869**	**0.869**	**0.869**	**0.869**	0.836	0.705	**0.869**
FreezerRegularTrain	0.986	0.986	0.956	**0.991**	0.989	0.922	0.899
Ham	0.714	0.733	0.724	**0.752**	0.743	0.524	0.467
MiddlePhalanxOutlineCorrect	**0.838**	0.821	0.825	0.818	0.818	0.753	0.698
DistalPhalanxOutlineCorrect	**0.761**	0.754	0.775	0.754	0.754	0.728	0.717
ProximalPhalanxOutlineCorrect	**0.887**	0.880	0.859	0.866	0.873	0.770	0.784
Mallat	0.914	0.869	**0.951**	0.871	0.922	0.713	0.934
InsectWingbeatSound	0.466	**0.630**	0.597	0.549	0.415	0.266	0.355
Rock	0.700	**0.780**	0.580	0.580	0.600	0.680	0.600
Average	**0.882**	**0.877**	**0.871**	**0.839**	**0.843**	**0.729**	**0.794**

Table 6. Average accuracy for time series classification on 50% of the UCR datasets. The best accuracy result for each dataset is shown in bold red font.

Dataset	H	univariate $alpha$	multivariate $alpha$
$ETTh_1$	24	2	10
	48	500	10
	168	1000	50
	336	1000	1000
	720	1000	1000
$ETTh_2$	24	0.1	50
	48	0.1	50
	168	0.1	1000
	336	0.1	1000
	720	0.1	1000
$ETTm_1$	24	0.2	5
	48	1	5
	96	20	2
	288	1000	5
	672	100	500
$Electric.$	24	0.1	N/A
	48	200	N/A
	96	200	N/A
	288	500	N/A
	672	1000	N/A

Table 7. Optimal $alpha$ values for the Ridge Regression estimator for the time series forecasting tasks. Note, the multivariate Electricity dataset timed out, therefore, we could not report its optimal values.

Dataset	Our TS2Vec Training Time (s)	TS2Vec Training Time (s)	Informer Training Time (s)
univariate $ETTh_1$	28.43	N/A	N/A
univariate $ETTh_2$	25.41	N/A	N/A
univariate $ETTm_1$	243.35	N/A	N/A
univariate $Electric.$	155.98	N/A	N/A
multivariate $ETTh_1$	73.72	N/A	N/A
multivariate $ETTh_2$	73.95	N/A	N/A
multivariate $ETTm_1$	246.89	345.44	2603.42
multivariate $Electric.$	N/A	N/A	N/A

Table 8. The total training time (s) for the time series forecasting tasks. Note, the multivariate Electricity dataset timed out, therefore, we could not report its training time. Yue et al. only provided the training time for TS2Vec and Informer on the multivariate $ETTm_1$ dataset. Our training time was faster than both TS2Vec and Informer.

Dataset	H	TS2Vec	Our TS2Vec	Informer	LogTrans	N-BEATS	TCN	LSTnet
Electric.	24	0.260	0.254	**0.251**	0.528	0.427	0.263	0.281
	48	0.319	**0.302**	0.346	0.409	0.551	0.373	0.381
	168	0.427	**0.421**	0.544	0.959	0.893	0.609	0.599
	336	0.565	**0.562**	0.713	1.079	1.035	0.855	0.823
	720	**0.861**	0.862	1.182	1.001	1.548	1.263	1.278
$ETTh_1$	24	**0.039**	0.045	0.098	0.103	0.094	0.075	0.108
	48	**0.062**	0.063	0.158	0.167	0.210	0.316	0.396
	168	0.134	**0.120**	0.183	0.207	0.232	0.316	0.396
	336	0.154	**0.140**	0.222	0.230	0.232	0.306	0.468
	720	0.163	**0.162**	0.269	0.273	0.322	0.390	0.659
$ETTh_2$	24	**0.090**	0.093	0.093	0.102	0.198	0.103	3.554
	48	**0.124**	**0.124**	0.155	0.169	0.234	0.142	3.190
	168	0.208	**0.197**	0232	0.246	0.331	0.227	2.800
	336	0.213	**0.199**	0.267	0.375	0.431	0.296	2.753
	720	**0.214**	**0.214**	0.277	0.303	0.437	0.325	2.878
$ETTm_1$	24	**0.015**	0.016	0.030	0.065	0.054	0.041	0.090
	48	**0.027**	0.031	0.069	0.078	0.190	0.101	0.179
	96	**0.044**	0.051	0.194	0.199	0.183	0.142	0.272
	288	0.103	**0.092**	0.401	0.411	0.186	0.318	0.462
	672	0.156	**0.130**	0.521	0.598	0.197	0.397	0.639
Average		**0.209**	**0.204**	**0.310**	**0.370**	**0.399**	**0.338**	**1.099**

Table 9. MSE results for the univariate time series forecasting task. The best MSE result for each dataset is shown in bold red font.

Dataset	H	TS2Vec	Our TS2Vec	Informer	LogTrans	N-BEATS	TCN	LSTnet
Electric.	24	0.288	0.291	**0.275**	0.447	0.330	0.279	0.287
	48	**0.324**	**0.324**	0.339	0.414	0.392	0.344	0.366
	168	0.394	**0.391**	0.424	0.612	0.538	0.462	0.500
	336	**0.474**	0.481	0.512	0.639	0.669	0.606	0.624
	720	**0.643**	0.652	0.806	0.714	0.881	0.858	0.906
$ETTh_1$	24	**0.152**	0.159	0.247	0.259	0.238	0.210	0.284
	48	0.191	**0.190**	0.319	0.328	0.367	0.402	0.424
	168	0.282	**0.265**	0.346	0.375	0.391	0.493	0.504
	336	0.310	**0.291**	0.387	0.398	0.388	0.495	0.593
	720	0.327	**0.323**	0.435	0.463	0.490	0.557	0.766
$ETTh_2$	24	**0.229**	0.233	0.240	0.255	0.345	0.249	0.445
	48	**0.273**	0.275	0.314	0.348	0.386	0.290	0.474
	168	0.360	**0.356**	0.389	0.422	0.453	0.376	0.595
	336	0.369	**0.359**	0.417	0.437	0.508	0.430	0.738
	720	**0.374**	0.375	0.431	0.493	0.517	0.463	1.044
$ETTm_1$	24	**0.092**	0.095	0.137	0.202	0.184	0.157	0.206
	48	**0.126**	0.132	0.203	0.220	0.361	0.257	0.306
	96	**0.161**	0.172	0.424	0.612	0.538	0.462	0.500
	288	0.246	**0.233**	0.512	0.639	0.669	0.606	0.624
	672	**0.278**	0.307	0.806	0.714	0.881	0.858	0.906
Average		**0.296**	**0.295**	**0.390**	**0.434**	**0.426**	**0.413**	**0.536**

Table 10. MAE results for the univariate time series forecasting task. The best MAE result for each dataset is shown in bold red font.

Dataset	H	TS2Vec	Our TS2Vec	Informer	StemGNN	TCN	LogTrans	LSTnet
$ETTh_1$	24	0.599	**0.521**	0.557	0.614	0.767	0.686	1.293
	48	0.629	**0.573**	0.685	0.748	0.713	0.766	1.456
	168	0.755	0.706	0.931	**0.663**	0.995	1.002	1.997
	336	0.907	**0.857**	1.128	0.927	1.175	1.362	2.655
	720	1.048	**1.029**	1.215	N/A	1.453	1.397	2.143
$ETTh_2$	24	**0.398**	0.402	0.720	1.292	1.365	0.828	2.742
	48	0.580	**0.577**	1.457	1.099	1.395	1.806	3.567
	168	1.901	**1.820**	3.489	2.282	3.166	4.070	3.242
	336	2.304	**2.215**	2.723	3.086	3.256	3.875	2.544
	720	**2.650**	2.660	3.467	N/A	3.690	3.913	4.625
$ETTm_1$	24	0.443	0.429	**0.323**	0.620	0.324	0.419	1.968
	48	0.582	0.554	0.494	0.744	**0.477**	0.507	1.999
	96	0.622	**0.587**	0.678	0.709	0.636	0.768	2.762
	288	0.709	**0.667**	1.056	0.843	1.270	1.462	1.257
	672	0.786	**0.754**	1.192	N/A	1.381	1.669	1.917
Average		**0.994**	**0.957**	**1.342**	**1.136**	**1.470**	**1.635**	**2.411**

Table 11. MSE results for the multivariate time series forecasting task. The best MSE result for each dataset is shown in bold red font.

Dataset	H	TS2Vec	Our TS2Vec	Informer	StemGNN	TCN	LogTrans	LSTnet
$ETTh_1$	24	0.534	**0.503**	0.549	0.571	0.612	0.604	0.901
	48	0.555	**0.536**	0.625	0.618	0.617	0.757	0.960
	168	0.636	0.614	0.752	**0.608**	0.738	0.846	1.214
	336	0.717	**0.695**	0.873	0.730	0.800	0.952	1.369
	720	0.790	**0.785**	0.896	N/A	1.311	1.291	1.380
$ETTh_2$	24	**0.461**	0.471	0.665	0.883	0.888	0.750	1.457
	48	**0.573**	0.582	1.001	0.847	0.960	1.034	1.687
	168	1.065	**1.064**	1.515	1.228	1.407	1.681	2.513
	336	1.215	**1.201**	1.340	1.351	1.481	1.763	2.591
	720	**1.373**	1.386	1.473	N/A	1.467	1.461	2.941
$ETTm_1$	24	0.436	0.436	**0.369**	0.570	0.374	0.412	1.170
	48	0.515	**0.503**	**0.503**	0.628	0.450	0.583	1.215
	96	0.549	**0.529**	0.614	0.624	0.602	0.792	1.542
	288	0.609	**0.581**	0.786	0.683	1.351	1.320	2.076
	672	0.655	**0.638**	0.926	N/A	1.467	1.461	2.941
Average		**0.712**	**0.702**	**0.859**	**0.778**	**0.968**	**1.045**	**1.730**

Table 12. MAE results for the multivariate time series forecasting task. The best MAE result for each dataset is shown in bold red font.

Parameter	Range	Number of Values
`alpha` (ridge-regression)	$0.1 - 1000$	27
`repr-dims`	$50 - 1280$	7
`epochs`	$200 - 800$	4
`lr`	$0.0001 - 0.1$	4
`max-training-length`	$500 - 8000$	7
`batch-size` (training)	$1, 8, 100$	3
`alpha` (loss function)	$0 - 1$	4
`padding`	$1 - 1000$	5
`sliding_length`	$1 - 1000$	5
`batch-size` (fit)	$100 - 1000$	4

Table 13. Scope of the hyperparameter search for groundwater depth dataset.