
Self-Distilled Representation Learning for Time Series

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 Self-supervised learning for time-series data holds potential similar to that re-
2 cently unleashed in Natural Language Processing and Computer Vision. While
3 most existing works in this area focus on contrastive learning, we propose a con-
4 ceptually simple yet powerful non-contrastive approach, based on the data2vec
5 self-distillation framework. The core of our method is a student-teacher scheme
6 that predicts the latent representation of an input time series from masked views of
7 the same time series. This strategy avoids strong modality-specific assumptions
8 and biases typically introduced by the design of contrastive sample pairs. We
9 demonstrate the competitiveness of our approach for classification and forecast-
10 ing as downstream tasks, comparing with state-of-the-art self-supervised learning
11 methods on the UCR and UEA archives as well as the ETT and Electricity datasets.

12 1 Introduction

13 Time series are a ubiquitous data resource in numerous application domains, ranging from finance
14 and healthcare to environmental monitoring and manufacturing. Understanding and harnessing their
15 inherent patterns is the driving force behind standard tasks like predictive analytics, forecasting, and
16 anomaly detection. Despite a notable body of work on *deep learning techniques* for time-series
17 analysis [17, 23, 31], more traditional methods like XGBoost [7] and handcrafted features continue to
18 play a pivotal role, often setting the gold standard in supervised learning and forecasting [18, 26, 34].
19 In particular, learning universal representations remains a fundamental challenge for temporal data.

20 Given recent breakthroughs in Natural Language Processing (NLP) and Computer Vision (CV), the
21 paradigm of *self-supervised learning (SSL)* has the potential to become a game changer in the area of
22 time series as well. While huge amounts of unlabeled temporal data exist in many business sectors, it
23 is fair to say that research in this direction and practical feasibility are still not mature.

24 A popular line of work on SSL for time series focuses on *contrastive methods*, aiming at robust data
25 representations by training neural networks to differentiate between positive (similar) and negative
26 (dissimilar) pairs of samples. Few prominent examples are TS2Vec [32], T-Loss [14], TS-TCC [13],
27 and TNC [30]; see [24, 36] for recent surveys of the field. Although the effectiveness of contrastive
28 methods has been demonstrated on several benchmark datasets, e.g., see [24], the design of positive
29 and negative samples for time series is not straightforward. Indeed, common augmentation strategies
30 from CV and NLP are not easily transferable, as modality-specific characteristics like temporal and
31 multi-scale dependencies need to be considered. As a consequence, the performance of existing
32 methods often strongly depends on the specific use case and task.

33 *Non-contrastive methods* are a promising remedy for addressing this lack of flexibility. As the name
34 suggests, this class of algorithms focuses on pretext tasks that encourage a model to learn meaningful
35 data representations without the explicit construction of positive and negative sample pairs. While
36 there exists a variety of non-contrastive SSL approaches in CV and NLP, e.g., see [2, 5, 6, 8, 15, 33],
37 they found much less attention in the time-series domain. In fact, existing work mostly focuses on

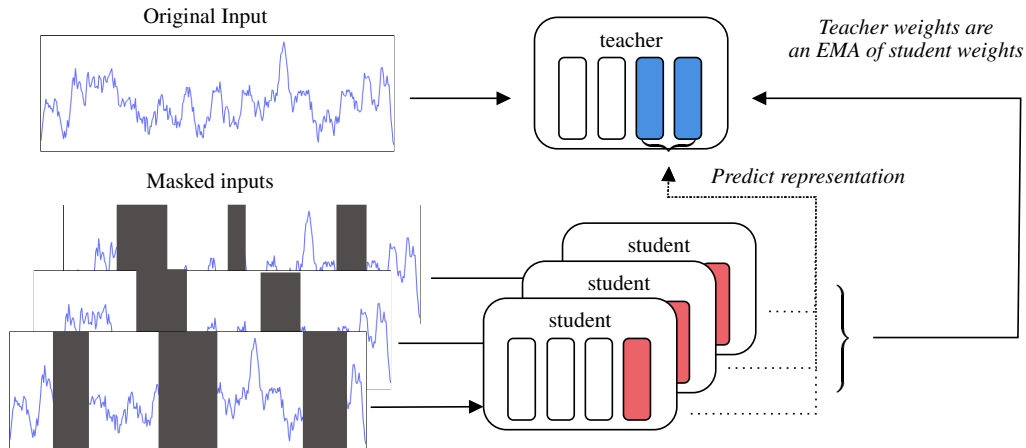


Figure 1: *Illustration of our data2vec-based training pipeline.* In teacher mode, the encoder computes a target representation of the full input time series by averaging the hidden activations of the last K encoder layers (shaded in blue). In student mode, this representation is then predicted by encoding (multiple) versions of the same input with randomly masked timestamps (shaded in red). As common in self-distillation schemes, the teacher’s weights follow the student’s weights according to an EMA.

38 “classical” unsupervised learning techniques like autoencoders [9, 22, 25], see [24, 36] for a broader
 39 overview.

40 The present work makes further progress on non-contrastive learning for time-series data and presents,
 41 to the best of our knowledge, the first method based on self-distillation. Our main contributions can
 42 be summarized as follows:

- 43 1. We propose a conceptually simple non-contrastive learning strategy for time-series data
 44 by adopting the recent *data2vec* framework [2]. The underlying idea is to leverage a
 45 student-teacher scheme to predict the *latent* representation of given input data based on a
 46 masked view of the same input. Unlike contrastive methods, no modality-specific designs
 47 are required in this process. On a larger scope, we underpin the main promise of *data2vec*
 48 (originally considered for vision, language, and speech) to provide a seamlessly extendable,
 49 modality-agnostic framework.
- 50 2. We demonstrate the effectiveness of our method for classification and forecasting down-
 51 stream tasks. In comparison with several existing SSL approaches, we report state-of-the-art
 52 performance on the UCR [10] & UEA [3] benchmark archives for classification as well as
 53 on the ETT [37] & Electricity [16] datasets for forecasting.

54 2 Methodology

55 **Self-distillation training objective.** Our training strategy closely follows the SSL approach of
 56 *data2vec* [2], which proposes a simple, yet effective self-distillation scheme. The *teacher model*
 57 provides a target representation of given input data, which the *student model* is supposed to predict
 58 from masked versions of the same input; see Figure 1 for an illustration. More specifically, the target
 59 representation is computed by averaging the hidden activations over the last K layers of the teacher
 60 model, which was found to stabilize the training dynamics [2]. Similarly to related self-distillation
 61 frameworks like BYOL [15] or DINO [6], the teacher’s weights follow the student model according
 62 to an *Exponential Moving Average (EMA)* mechanism during training. We argue that *data2vec* is
 63 well-suited for our purposes because of its simplicity and generalizability. Our simple timestamp
 64 masking strategy particularly bypasses the limitations and unintentional biases that typically occur
 65 when handcrafting positive and negative samples in contrastive methods.

Table 1: Summarized results for time-series classification using accuracy as metric. We report the average scores over all datasets of each archive (128 datasets for UCR and 30 datasets for UEA, respectively). See Table 3 and 4 in Appendix C for the full results.

	Ours	TS2Vec [32]	Ti-MAE [22]	T-Loss [14]	TNC [30]	TS-TCC [13]	TST [35]
UCR	0.832	0.829	0.823	0.806	0.761	0.757	0.638
UEA	0.738	0.704	-	0.658	0.670	0.668	0.617

66 For a more detailed introduction to data2vec and an in-depth analysis of its design choices, we refer
 67 to the original paper [2] as well as its successor data2vec 2.0 [1]. Moreover, we point out some
 68 differences between our approach and the original framework in Appendix A.

69 **Encoder architecture.** A notable difference from the original data2vec scheme is our choice of
 70 encoder backbone: instead of a transformer-based model, we employ a *Convolutional Neural Network*
 71 (*CNN*). This design choice aligns our approach closer with existing (contrastive) SSL methods for
 72 time-series data and allows for a more direct comparison. Our specific architecture is inspired by
 73 the *TS2Vec* encoder [32], which proposes a cascade of residual convolutional blocks. Here, the l -th
 74 block applies two consecutive 1D convolutions with dilation parameter 2^l to enlarge the receptive
 75 field over the temporal axis. At the same time, a suitable padding scheme ensures consistent feature
 76 dimensionality from layer to layer, which is key to the hierarchical contrastive loss function developed
 77 in [32]. Our learning protocol also exploits this consistency albeit in a different way, namely by
 78 computing the averaged target representation vector over the last K layers. In contrast to the original
 79 TS2Vec architecture, we have incorporated batch normalization after each convolutional layer as well
 80 as a tunable scaling factor for the weight initialization, both of which enhanced the stability of our
 81 self-distillation training pipeline.

82 Finally, let us emphasize that the representations produced by our CNN encoder are still sequential,
 83 i.e., one feature vector is computed per timestamp. While this is analogous to transformer-based
 84 encoders, our feature vectors get “contextualized” by exponentially increasing the dilation parameter
 85 instead of using self-attention layers.

86 3 Experimental Results

87 We assess the effectiveness of our method with respect to its downstream task performance in
 88 time-series classification and forecasting. Our basic experimental setup follows a simple two-step
 89 procedure for each considered dataset: (1) learning the encoder in a self-supervised fashion without
 90 any labels, and (2) training a task-specific layer on top of the learned representations, while the
 91 encoder’s weights are frozen. This protocol is closely aligned with the one of TS2Vec [32], which
 92 will serve as our primary reference point for comparison with state-of-the-art SSL methods;¹ see
 93 also [24] for an independent benchmarking study.

94 It is well-known that self-distillation is prone to representation collapse, which is why we performed
 95 a preliminary hyperparameter optimization (HPO) on a small subset of the UCR archive to select
 96 important training parameters like the learning rate or EMA parameters. In the actual experiments,
 97 all hyperparameters (including the CNN encoder architecture, which is not tuned) remain fixed and
 98 consistent. For more details on the experimental setup and implementation, see Appendix A.

99 **Time-series classification.** In this standard downstream task, each time-series instance is associated
 100 with a single label to be predicted. To obtain instance-level representations, we first perform a max-
 101 aggregation over all timestamps. The resulting (fixed-size) feature vector is then used as input for an
 102 SVM head with RBF kernel, which is trained on the labeled dataset. Following [32], we benchmark
 103 our approach on the UCR archive [10] and UEA archive [3], which consist of 128 (univariate) and 30
 104 (multivariate) datasets, respectively.

105 Our experimental results are summarized in Table 1. For the UCR archive, we have also included
 106 the scores reported for Ti-MAE [22], which is a recent non-contrastive approach based on a masked
 107 autoencoder. We conclude that our data2vec scheme is highly competitive with existing SSL methods,

¹The reported scores for all comparison methods—not only TS2Vec—are taken from [32]; see Appendix B for more details.

Table 2: Summarized results for time-series forecasting using mean squared error (MSE) and mean absolute error (MAE) as metrics. For each dataset, we report the average scores over all values of H (= number of future observations to be predicted). See Table 5 and 6 in Appendix C for the full results, including more comparison methods.

Dataset	Ours		TS2Vec [32]		Informer [37]		LogTrans [21]		TCN [4]		
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	
Univariate	ETTh ₁	0.1303	0.2744	0.1104	0.2524	0.186	0.3468	0.196	0.3646	0.2628	0.4314
	ETTh ₂	0.1445	0.2944	0.1698	0.321	0.204	0.3582	0.2174	0.391	0.2186	0.3616
	ETTM ₁	0.0741	0.1952	0.069	0.1864	0.2412	0.382	0.2702	0.4164	0.1998	0.3488
	Electricity	0.3263	0.4243	0.4864	0.4246	0.6072	0.4712	0.7952	0.5652	0.6726	0.5098
	Avg.	0.1688	0.2971	0.209	0.296	0.31	0.39	0.37	0.434	0.338	0.413
Multivariate	ETTh1	0.667	0.616	0.788	0.646	0.907	0.739	1.043	0.89	1.021	0.816
	ETTh2	0.716	0.65	1.567	0.937	2.371	1.199	2.898	1.356	2.574	1.265
	ETTM1	0.506	0.522	0.628	0.553	0.749	0.64	0.965	0.914	0.818	0.849
	Electricity	0.297	0.392	0.33	0.405	0.589	0.548	0.35	0.41	0.355	0.42
	Avg.	0.546	0.545	0.828	0.636	1.154	0.781	1.314	0.892	1.192	0.837

108 slightly outperforming TS2Vec on UCR and even more clearly on UEA. Here, the UEA archive can
 109 be considered more challenging due to its multivariate nature.

110 **Time-series forecasting.** Given a time series up to a certain timestamp, forecasting aims to
 111 predict future observations. Our downstream protocol first extracts the last encoded feature vector
 112 (corresponding to the last observed timestamp), which is then used as input to train a ridge regression
 113 head that predicts the next H observations. Adopting the experimental setup of [32] again, we
 114 consider three versions of the ETT datasets [37] as well as the Electricity dataset [16], both in the uni-
 115 and multivariate setting.

116 Our results are summarized in Table 2. While our data2vec approach is consistently competitive in
 117 the univariate case, we highlight a notable improvement in MSE on Electricity and ETTh₂. Similarly
 118 to classification, the performance gain becomes even more striking in the multivariate case, for which
 119 we report superior results across almost all datasets.

120 4 Discussion

121 This work provides initial evidence for the effectiveness of SSL via self-distillation in the time-
 122 series domain. Our experimental study particularly demonstrates that state-of-the-art performance in
 123 classification and forecasting is achievable without strong modality-specific assumptions, which are
 124 typically made by contrastive methods.

125 **Scope and limitations.** Despite competitive empirical results, the scope of our work is linked
 126 to the limitations of the considered benchmark archives. Although these datasets are diverse and
 127 widely used in the related literature, we believe that they are not perfectly suited for an assessment of
 128 large-scale (deep-)learning methods, especially SSL and pre-training techniques. For example, the
 129 UCR archive contains rather small datasets, some of which have quite degenerated train/test splits,
 130 resulting in noisy and insignificant evaluations regardless of the used learning algorithm. A specific
 131 limitation of our self-distillation framework is its sensitivity to the training parameters. In fact, to
 132 produce robust representations and prevent model collapses, additional hyperparameter tuning is
 133 required in advance.

134 **Outlook and challenges.** Obvious avenues of future research are the exploration of other non-
 135 contrastive methods as well as different types of encoder backbones. In the bigger picture, we argue
 136 that large-scale experiments are indispensable to unleash and certify the power of (SSL) deep-learning
 137 methods for time-series analysis. To catch up with the more mature fields of CV and NLP, perhaps
 138 the most important challenge is the creation of large, inhomogeneous cohorts of (publicly available)
 139 time-series data; see TimeGPT [27] for a very recent effort in that direction. Beyond that, we believe
 140 that more fundamental modality-specific research is required for future breakthroughs. For instance,
 141 temporal data still lacks a unified tokenization strategy, unlike NLP and CV where well-established
 142 tokenizers are crucial to the current success of Large Language Models and Vision Transformers [12].

143 **References**

- 144 [1] A. Baevski, A. Babu, W.-N. Hsu, and M. Auli. Efficient self-supervised learning with contextualized target
 145 representations for vision, speech and language. In *Proceedings of the 40th International Conference on*
 146 *Machine Learning (ICML)*, pages 1416–1429, 2023.
- 147 [2] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli. data2vec: A general framework for self-
 148 supervised learning in speech, vision and language. In *Proceedings of the 39th International Conference*
 149 *on Machine Learning (ICML)*, pages 1298–1312, 2022.
- 150 [3] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. The UEA
 151 multivariate time series classification archive, 2018. Preprint arXiv:1811.00075, 2018.
- 152 [4] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks
 153 for sequence modeling. Preprint arXiv:1803.01271, 2018.
- 154 [5] A. Bardes, J. Ponce, and Y. LeCun. VICReg: Variance-invariance-covariance regularization for self-
 155 supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.
- 156 [6] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties
 157 in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on*
 158 *Computer Vision (ICCV)*, pages 9650–9660, 2021.
- 159 [7] T. Chen and C. Guestrin. XGBoost: a scalable tree boosting system. In *Proceedings of the 22nd ACM*
 160 *SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
- 161 [8] X. Chen and K. He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF*
 162 *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, 2021.
- 163 [9] M. Cheng, Q. Liu, Z. Liu, H. Zhang, R. Zhang, and E. Chen. TimeMAE: Self-Supervised Representations
 164 of Time Series with Decoupled Masked Autoencoders. Preprint arXiv:2303.00320, 2023.
- 165 [10] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and
 166 E. Keogh. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.
- 167 [11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers
 168 for language understanding. Preprint arXiv:1810.04805, 2018.
- 169 [12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Min-
 170 derer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers
 171 for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.
- 172 [13] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. Keong, X. L. Kwoh, and C. Guan. Time-series representation
 173 learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint*
 174 *Conference on Artificial Intelligence (IJCAI-21)*, 2021.
- 175 [14] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi. Unsupervised scalable representation learning for multivari-
 176 ate time series. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019.
- 177 [15] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires,
 178 Z. Guo, M. Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning.
 179 In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.
- 180 [16] G. Hebrail and A. Berard. Individual household electric power consumption. UCI Machine Learning
 181 Repository, <https://doi.org/10.24432/C58K54>, 2012.
- 182 [17] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series
 183 classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.
- 184 [18] T. Januschowski, Y. Wang, K. Torkkola, T. Erkkilä, H. Hasson, and J. Gasthaus. Forecasting with trees.
 185 *International Journal of Forecasting*, 38(4):1473–1481, 2022.
- 186 [19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. Preprint arXiv:1412.6980, 2014.
- 187 [20] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. Modeling long- and short-term temporal patterns with deep
 188 neural networks. In *The 41st International ACM SIGIR Conference on Research and Development in*
 189 *Information Retrieval*, 2018.
- 190 [21] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan. Enhancing the locality and breaking
 191 the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information*
 192 *Processing Systems 32 (NeurIPS)*, 2019.
- 193 [22] Z. Li, Z. Rao, L. Pan, P. Wang, and Z. Xu. Ti-MAE: self-supervised masked time series autoencoders.
 194 Preprint arXiv:2301.08871, 2023.
- 195 [23] B. Lim and S. Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of*
 196 *the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209, 2021.
- 197 [24] Q. Ma, Z. Liu, Z. Zheng, Z. Huang, S. Zhu, Z. Yu, and J. T. Kwok. A survey on time-series pre-trained
 198 models. Preprint arXiv:2305.10716, 2023.

- 199 [25] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff. Timenet: Pre-trained deep recurrent neural network
200 for time series classification. Preprint arXiv:1706.08838, 2017.
- 201 [26] Nixtla Inc. Statistical vs deep learning forecasting methods. [https://github.com/Nixtla/
202 statsforecast/tree/main/experiments/m3](https://github.com/Nixtla/statsforecast/tree/main/experiments/m3).
- 203 [27] Nixtla Inc. TimeGPT Beta. <https://docs.nixtla.io/>.
- 204 [28] B. N. Oreshkin, D. Carпов, N. Chapados, and Y. Bengio. N-BEATS: neural basis expansion analysis for
205 interpretable time series forecasting. In *International Conference on Learning Representations (ICLR)*,
206 2020.
- 207 [29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein,
208 L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner,
209 L. Fang, J. Bai, and S. Chintala. PyTorch: an imperative style, high-performance deep learning library.
210 Preprint arXiv:1912.01703, 2019.
- 211 [30] S. Tonekaboni, D. Eytan, and A. Goldenberg. Unsupervised representation learning for time series with
212 temporal neighborhood coding. In *International Conference on Learning Representations (ICLR)*, 2021.
- 213 [31] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey. In
214 *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023)*, 2022.
- 215 [32] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu. TS2Vec: towards universal representation
216 of time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8980–8987, 2022.
- 217 [33] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow Twins: self-supervised learning via redundancy
218 reduction. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages
219 12310–12320, 2021.
- 220 [34] A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are transformers effective for time series forecasting? *Proceedings
221 of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, 2023.
- 222 [35] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff. A transformer-based framework for
223 multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD International
224 Conference on Knowledge Discovery and Data Mining*, 2021.
- 225 [36] K. Zhang, Q. Wen, C. Zhang, R. Cai, M. Jin, Y. Liu, J. Zhang, Y. Liang, G. Pang, D. Song, and S. Pan. Self-
226 supervised learning for time series analysis: Taxonomy, progress, and prospects. Preprint arXiv:2306.10125,
227 06 2023.
- 228 [37] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient
229 transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial
230 Intelligence*, 35(12):11106–11115, 2021.

231 A Additional Implementation Details

232 This part complements Section 3 and describes more details on the implementation of our experiments
233 as well as some specific design choices.

234 **Datasets.** All considered datasets are accompanied by predefined splits, which we adopt to ensure
235 direct comparability with the other methods.

236 The UCR archive [10] has established itself as a standard benchmark in time-series classification,
237 providing a collection of 128 univariate datasets from various fields such as finance, healthcare, and
238 climatology. The UEA archive [3] is similarly diverse with the important difference that it contains
239 multivariate datasets, and can therefore be seen as more challenging.

240 The ETT datasets [37] provide hourly energy consumption metrics and are widely used as a forecasting
241 benchmark. We consider the versions ETTh₁, ETTh₂, and ETTm₁ in the uni- and multivariate case.
242 The Electricity dataset [16] records high-frequency household electricity consumption data. Its size
243 and complexity make it a suitable testbed for time-series representation learning techniques as well.

244 **Downstream evaluation.** For time-series classification, we follow the experimental setup of [32],
245 which is based on the standard SVM implementation of `scikit-learn`, specified to an RBF kernel
246 and a one-vs-rest strategy for multiclass classification; each evaluation step involves a simple grid
247 search cross-validation to optimize for the SVM regularization parameter C . We perform downstream
248 evaluations at regular intervals during pre-training to assess the quality of our learned representations.
249 The validation accuracy of the best evaluation then yields the final performance score.

250 Our approach to time-series forecasting is analogous. Here, we use the standard ridge regression
251 module of `scikit-learn`, tuning the regularization strength α through a grid search cross-
252 validation in each downstream evaluation.

253 **Hyperparameters and pre-training.** Across all experiments, we use the Adam optimizer [19]
254 and set the training batch size to 8. Following `data2vec`, we use a Smooth L1-loss to measure the
255 distance between the teacher’s representation targets and the student’s predictions.

256 For the pre-training phase, we ensure that each dataset undergoes an equivalent number of time steps.
257 This means that the total number of training steps is proportional to the length of the time series. We
258 also include a warmup phase, using the OneCycle learning rate scheduler to prevent overfitting on
259 local minima and to allow sufficient time for the batch normalization to adjust. To address the higher
260 dimensionality of the UEA datasets, we crop each input time series to a random window of size 1024.
261 These windows are selected independently for each sample and every training iteration. Similarly, for
262 forecasting, we use a cropping window size of 200, which is consistent with `TS2Vec`.

263 To stabilize our self-distillation approach, some important training parameters are selected through a
264 preliminary HPO. As auxiliary validation metric, we apply our representation learning method to
265 8 UCR datasets and measure the average accuracy achieved by training a simple logistic regression
266 classification head. The tuned hyperparameters are as follows: learning rate & scheduler warm-up
267 parameter, weight decay, EMA parameters, block masking probability, encoder dropout rate, scaling
268 factor for the random weight initialization of the encoder, and the β -parameter of the Smooth L1-loss.
269 The selected parameters are used consistently over all experiments described in Section 3.

270 All experiments were conducted on a Kubernetes Cluster hosted on Google Cloud Platform, using
271 NVIDIA Tesla T4 GPU accelerators. `PyTorch` [29] and `Lightning` are used as underlying deep
272 learning framework.

273 **data2vec and CNN encoder.** Compared to the original `data2vec` framework [2], we also incor-
274 porate some extensions from `data2vec 2.0` [1]. The first adaptation is a block masking algorithm,
275 which is a simplification of the inverse masking technique proposed by `data2vec 2.0`. Our approach
276 iterates through each (student) batch of time series data until the cumulative proportion of masked
277 blocks marginally exceeds a predefined masking probability. In every iteration step, we inject a new
278 masked block into each time series, where the size and location of this block are randomly selected
279 and bounded by the missing blocks. This ensures that each time series of a batch has masked blocks
280 that vary in size and position, thereby enhancing the robustness of the representation learning process.
281 Note that the masking probability was tuned through our preliminary HPO.

282 A second notable adoption from data2vec 2.0 is the use of multiple student representations to amortize
283 the costs of the teacher model computation. In our experiments, the number of students is consistently
284 set to 3.

285 We update the teacher weights according to an EMA:

$$w_{\text{teacher}} \leftarrow (1 - \delta) \cdot w_{\text{student}} + \delta \cdot w_{\text{teacher}}.$$

286 The update parameter δ starts at 0.9996 and linearly increases to 0.99996 over the training process.
287 These choices were proposed by our preliminary HPO, but it is noteworthy that the numerical values
288 differ only in the least significant digits from the ones reported in data2vec [2].

289 For the CNN encoder described in Section 2, we use 7 residual convolutional layers, which equals
290 the number of layers K over which the data2vec teacher model computes its representation, i.e., all
291 hidden encoder blocks are used for the averaging. The feature dimension in the representation space
292 is set to 320 (per timestamp), which equals the choice of TS2Vec.

293 B Comparison Methods

294 Below, we briefly describe all comparison methods considered in Section 3. Our selection is adopted
295 from [32], which is also the origin of the scores reported in this work (except for our method and
296 Ti-MAE [22]). We refer to [32] for reproduction details of each method as well as a more extensive
297 discussion of conceptual differences between them.

298 Comparison methods for classification:

- 299 • TS2Vec [32] proposes a contrastive method that learns contextual representations based
300 on a hierarchical loss that considers contrast on multiple resolution scales. Here, negative
301 samples are obtained both instance-wise and on the temporal axis. Positive samples are
302 generated through contextual consistency of augmented views of the input time series.
- 303 • Ti-MAE [22] introduces a non-contrastive representation learning approach, which randomly
304 masks parts of an embedded time series and learns to reconstruct it through an autoencoder
305 scheme. Both the encoder and decoder are based on transformer blocks.
- 306 • TNC [30] proposes a contrastive learning approach that exploits the local smoothness of time-
307 series signals to define neighborhoods over the temporal axis. Their contrastive loss intends
308 to distinguish the encoded representations of neighborhood signals from non-neighborhood
309 signals.
- 310 • TS-TCC [13] leverages both temporal and contextual contrasting, encouraging similarity
311 among different contexts of the same time-series sample while minimizing similarity among
312 contexts of different samples. Weak and strong augmentations are used to generate different
313 yet correlated views.
- 314 • T-Loss [14] proposes a representation learning approach based on a triplet loss and time-
315 based negative sampling. Here, random sub-time series are used to design positive pairs,
316 while different time-series instances are used as negative pairs.
- 317 • TST [35] uses a transformer-based model for pre-training on multivariate time series. Their
318 training objective is inspired by BERT-style models [11], predicting a time-series signal
319 from a randomly masked version thereof.

320 Comparison methods for forecasting:

- 321 • Informer [37] is a transformer-based model specifically designed for long-sequence time-
322 series forecasting. It proposes an efficient probabilistic attention mechanism achieving
323 $O(L \log L)$ complexity in time and memory, thereby avoiding the well-known quadratic
324 bottleneck of standard attention modules.
- 325 • LogTrans [21] proposes the LogSparse Transformer architecture, which is based on a
326 convolutional self-attention block that enhances the incorporation of local context. In this
327 way, they achieve super-linear memory complexity, similarly to the Informer.
- 328 • The authors of TCN [4] conducted a systematic experimental study of generic convolutional
329 and recurrent architectures for sequence modeling. They find that a simple Temporal

330 Convolutional Network (TCN) outperforms common recurrent architectures such as LSTMs
331 on various tasks and datasets.

- 332 • LSTnet [20] leverages a combination of Convolutional and Recurrent Neural Networks that
333 takes into account both short-term local dependencies and long-term trends in sequential
334 data.
- 335 • N-BEATS [28] proposes a deep stack of multi-layer fully connected blocks with forward
336 and backward residual connections. A particular feature of their network design is that its
337 outputs are human-interpretable.

338 **C Full Experimental Results**

339 The full results on all datasets of the UCR and UEA archive are reported in Table 3 and 4, respectively.
340 Table 5 and 6 show the full results of our forecasting experiments in the uni- and multivariate case,
341 respectively.

Table 3: Full results for time-series classification on the **UCR** archive using accuracy as metric. The reported scores for Ti-MAE are taken from [22] and the scores for all other comparison methods are taken from [32]. Note that for each method, pre-training and the downstream task are performed for each dataset individually.

UCR Dataset	Ours	TS2Vec	Ti-MAE	T-Loss	TNC	TS-TCC	TST
Avg. acc.	0.832	0.829	0.823	0.806	0.761	0.757	0.638
Avg. rank	2.109	2.586	2.656	3.720	4.584	4.512	6.234
ACSF1	0.830	0.900	0.820	0.900	0.730	0.730	0.760
Adiac	0.788	0.762	0.788	0.675	0.726	0.767	0.550
AllGestureWiimoteX	0.776	0.777	0.633	0.763	0.703	0.697	0.259
AllGestureWiimoteY	0.777	0.793	0.682	0.726	0.699	0.741	0.423
AllGestureWiimoteZ	0.749	0.746	0.671	0.723	0.646	0.689	0.447
ArrowHead	0.857	0.857	0.874	0.766	0.703	0.737	0.771
BME	1.000	0.993	1.000	0.993	0.973	0.933	0.760
Beef	0.667	0.767	0.900	0.667	0.733	0.600	0.500
BeetleFly	0.950	0.900	0.900	0.800	0.850	0.800	1.000
BirdChicken	0.800	0.800	1.000	0.850	0.750	0.650	0.650
CBF	0.997	1.000	1.000	0.983	0.983	0.998	0.898
Car	0.850	0.833	0.867	0.833	0.683	0.583	0.550
Chinatown	0.965	0.965	0.985	0.951	0.977	0.983	0.936
ChlorineConcentration	0.754	0.832	0.725	0.749	0.760	0.753	0.562
CinCECGTorso	0.654	0.827	0.971	0.713	0.669	0.671	0.508
Coffee	1.000	1.000	1.000	1.000	1.000	1.000	0.821
Computers	0.772	0.660	0.780	0.664	0.684	0.704	0.696
CricketX	0.767	0.782	0.674	0.713	0.623	0.731	0.385
CricketY	0.751	0.749	0.659	0.728	0.597	0.718	0.467
CricketZ	0.754	0.792	0.718	0.708	0.682	0.713	0.403
Crop	0.763	0.756	0.751	0.722	0.738	0.742	0.710
DiatomSizeReduction	0.987	0.984	0.984	0.984	0.993	0.977	0.961
DistalPhalanxOutlineAgeGroup	0.755	0.727	0.763	0.727	0.741	0.755	0.741
DistalPhalanxOutlineCorrect	0.804	0.761	0.793	0.775	0.754	0.754	0.728
DistalPhalanxTW	0.748	0.698	0.727	0.676	0.669	0.676	0.568
DodgerLoopDay	0.613	0.562	0.613	–	–	–	0.200
DodgerLoopGame	0.913	0.841	0.739	–	–	–	0.696
DodgerLoopWeekend	0.978	0.964	0.978	–	–	–	0.732
ECG200	0.940	0.920	0.910	0.940	0.830	0.880	0.830
ECG5000	0.940	0.935	0.942	0.933	0.937	0.941	0.928
ECGFiveDays	1.000	1.000	0.988	1.000	0.999	0.878	0.763
EOGHorizontalSignal	0.608	0.539	0.558	0.605	0.442	0.401	0.373
EOGVerticalSignal	0.475	0.503	0.547	0.434	0.392	0.376	0.298
Earthquakes	0.748	0.748	0.748	0.748	0.748	0.748	0.748
ElectricDevices	0.704	0.721	0.685	0.707	0.700	0.686	0.676
EthanolLevel	0.670	0.468	0.744	0.382	0.424	0.486	0.260
FaceAll	0.873	0.771	0.880	0.786	0.766	0.813	0.504
FaceFour	0.795	0.932	0.875	0.920	0.659	0.773	0.511
FacesUCR	0.908	0.924	0.866	0.884	0.789	0.863	0.543
FiftyWords	0.769	0.771	0.787	0.732	0.653	0.653	0.525
Fish	0.937	0.926	0.897	0.891	0.817	0.817	0.720

(continued on next page)

Table 3: Full results for time-series classification on the UCR archive (continued from previous page).

UCR Dataset	Ours	TS2Vec	Ti-MAE	T-Loss	TNC	TS-TCC	TST
FordA	0.930	0.936	0.818	0.928	0.902	0.930	0.568
FordB	0.790	0.794	0.652	0.793	0.733	0.815	0.507
FreezerRegularTrain	0.998	0.986	0.987	0.956	0.991	0.989	0.922
FreezerSmallTrain	0.980	0.870	0.959	0.933	0.982	0.979	0.920
Fungi	0.989	0.957	0.968	1.000	0.527	0.753	0.366
GestureMidAirD1	0.654	0.608	0.662	0.608	0.431	0.369	0.208
GestureMidAirD2	0.631	0.469	0.546	0.546	0.362	0.254	0.138
GestureMidAirD3	0.331	0.292	0.400	0.285	0.292	0.177	0.154
GesturePebbleZ1	0.738	0.930	0.901	0.919	0.378	0.395	0.500
GesturePebbleZ2	0.677	0.873	0.918	0.899	0.316	0.430	0.380
GunPoint	1.000	0.980	0.993	0.980	0.967	0.993	0.827
GunPointAgeSpan	0.994	0.987	0.994	0.994	0.984	0.994	0.991
GunPointMaleVersusFemale	1.000	1.000	0.997	0.997	0.994	0.997	1.000
GunPointOldVersusYoung	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Ham	0.733	0.714	0.800	0.724	0.752	0.743	0.524
HandOutlines	0.916	0.922	0.919	0.922	0.930	0.724	0.735
Haptics	0.464	0.526	0.484	0.490	0.474	0.396	0.357
Herring	0.641	0.641	0.656	0.594	0.594	0.594	0.594
HouseTwenty	0.941	0.916	0.941	0.933	0.782	0.790	0.815
InlineSkate	0.471	0.415	0.380	0.371	0.378	0.347	0.287
InsectEPGRegularTrain	1.000	1.000	1.000	1.000	1.000	1.000	1.000
InsectEPGSmallTrain	1.000	1.000	1.000	1.000	1.000	1.000	1.000
InsectWingbeatSound	0.590	0.630	0.639	0.597	0.549	0.415	0.266
ItalyPowerDemand	0.963	0.925	0.967	0.954	0.928	0.955	0.845
LargeKitchenAppliances	0.861	0.845	0.787	0.789	0.776	0.848	0.595
Lightning2	0.902	0.869	0.836	0.869	0.869	0.836	0.705
Lightning7	0.808	0.863	0.808	0.795	0.767	0.685	0.411
Mallat	0.950	0.914	0.956	0.951	0.871	0.922	0.713
Meat	0.967	0.950	0.967	0.950	0.917	0.883	0.900
MedicalImages	0.803	0.789	0.771	0.750	0.754	0.747	0.632
MelbournePedestrian	0.958	0.959	0.949	0.944	0.942	0.949	0.741
MiddlePhalanxOutlineAgeGroup	0.649	0.636	0.675	0.656	0.643	0.630	0.617
MiddlePhalanxOutlineCorrect	0.852	0.838	0.811	0.825	0.818	0.818	0.753
MiddlePhalanxTW	0.623	0.584	0.623	0.591	0.571	0.610	0.506
MixedShapes	0.922	0.917	0.922	0.905	0.911	0.855	0.879
MixedShapesSmallTrain	0.877	0.861	0.875	0.860	0.813	0.735	0.828
MoteStrain	0.880	0.861	0.913	0.851	0.825	0.843	0.768
NonInvasiveFetalECGThorax1	0.924	0.930	0.918	0.878	0.898	0.898	0.471
NonInvasiveFetalECGThorax2	0.930	0.938	0.938	0.919	0.912	0.913	0.832
OSULeaf	0.806	0.851	0.736	0.760	0.723	0.723	0.545
OliveOil	0.867	0.900	0.933	0.867	0.833	0.800	0.800
PLAID	0.449	0.561	0.458	0.555	0.495	0.445	0.419
PhalangesOutlinesCorrect	0.834	0.809	0.772	0.784	0.787	0.804	0.773
Phoneme	0.266	0.312	0.229	0.276	0.180	0.242	0.139
PickupGestureWiimoteZ	0.700	0.820	0.840	0.740	0.620	0.600	0.240

(continued on next page)

Table 3: Full results for time-series classification on the UCR archive (continued from previous page).

UCR Dataset	Ours	TS2Vec	Ti-MAE	T-Loss	TNC	TS-TCC	TST
PigAirwayPressure	0.793	0.630	0.240	0.510	0.413	0.380	0.120
PigArtPressure	0.904	0.966	0.760	0.928	0.808	0.524	0.774
PigCVP	0.889	0.812	0.750	0.788	0.649	0.615	0.596
Plane	1.000	1.000	1.000	0.990	1.000	1.000	0.933
PowerCons	0.961	0.961	1.000	0.900	0.933	0.961	0.911
ProximalPhalanxOutlineAgeGroup	0.883	0.834	0.863	0.844	0.854	0.839	0.854
ProximalPhalanxOutlineCorrect	0.883	0.887	0.876	0.859	0.866	0.873	0.770
ProximalPhalanxTW	0.824	0.824	0.829	0.771	0.810	0.800	0.780
RefrigerationDevices	0.571	0.589	0.611	0.515	0.565	0.563	0.483
Rock	0.840	0.700	0.660	0.580	0.580	0.600	0.680
ScreenType	0.480	0.411	0.579	0.416	0.509	0.419	0.419
SemgHandGenderCh2	0.900	0.963	0.838	0.890	0.882	0.837	0.725
SemgHandMovementCh2	0.713	0.860	0.700	0.789	0.593	0.613	0.420
SemgHandSubjectCh2	0.813	0.951	0.813	0.853	0.771	0.753	0.484
ShakeGestureWiimoteZ	0.900	0.940	0.900	0.920	0.820	0.860	0.760
ShapeletSim	1.000	1.000	0.911	0.672	0.589	0.683	0.489
ShapesAll	0.855	0.902	0.840	0.848	0.788	0.773	0.733
SmallKitchenAppliances	0.699	0.731	0.741	0.677	0.725	0.691	0.592
SmoothSubspace	1.000	0.980	0.993	0.960	0.913	0.953	0.827
SonyAIBORobotSurface1	0.918	0.903	0.912	0.902	0.804	0.899	0.724
SonyAIBORobotSurface2	0.858	0.871	0.934	0.889	0.834	0.907	0.745
StarLightCurves	0.979	0.969	0.972	0.964	0.968	0.967	0.949
Strawberry	0.978	0.962	0.970	0.954	0.951	0.965	0.916
SwedishLeaf	0.962	0.941	0.938	0.914	0.880	0.923	0.738
Symbols	0.971	0.976	0.961	0.963	0.885	0.916	0.786
SyntheticControl	1.000	0.997	0.993	0.987	1.000	0.990	0.490
ToeSegmentation1	0.947	0.917	0.890	0.939	0.864	0.930	0.807
ToeSegmentation2	0.908	0.892	0.908	0.900	0.831	0.877	0.615
Trace	1.000	1.000	1.000	0.990	1.000	1.000	1.000
TwoLeadECG	0.999	0.986	0.985	0.999	0.993	0.976	0.871
TwoPatterns	1.000	1.000	0.994	0.999	1.000	0.999	0.466
UMD	1.000	1.000	1.000	0.993	0.993	0.986	0.910
UWaveGestureLibraryAll	0.878	0.930	0.956	0.896	0.903	0.692	0.475
UWaveGestureLibraryX	0.823	0.795	0.814	0.785	0.781	0.733	0.569
UWaveGestureLibraryY	0.762	0.719	0.736	0.710	0.697	0.641	0.348
UWaveGestureLibraryZ	0.769	0.770	0.749	0.757	0.721	0.690	0.655
Wafer	0.998	0.998	0.996	0.992	0.994	0.994	0.991
Wine	0.944	0.870	0.907	0.815	0.759	0.778	0.500
WordSynonyms	0.699	0.676	0.705	0.691	0.630	0.531	0.422
Worms	0.792	0.701	0.779	0.727	0.623	0.753	0.455
WormsTwoClass	0.844	0.805	0.792	0.792	0.727	0.753	0.584
Yoga	0.872	0.887	0.834	0.837	0.812	0.791	0.830

Table 4: Full results for time-series classification on the **UEA** archive using accuracy as metric. The reported scores for all comparison methods are taken from [32]. Note that for each method, pre-training and the downstream task are performed for each dataset individually.

UEA Dataset	Ours	TS2Vec	T-Loss	TNC	TS-TCC	TST
Avg. acc.	0.738	0.704	0.658	0.670	0.668	0.617
Avg. rank	1.700	2.733	3.500	4.033	3.833	4.633
ArticularyWordRecognition	0.990	0.987	0.943	0.973	0.953	0.977
AtrialFibrillation	0.267	0.200	0.133	0.133	0.267	0.067
BasicMotions	0.925	0.975	1.000	0.975	1.000	0.975
CharacterTrajectories	0.994	0.995	0.993	0.967	0.985	0.975
Cricket	1.000	0.972	0.972	0.958	0.917	1.000
DuckDuckGeese	0.480	0.680	0.650	0.460	0.380	0.620
EigenWorms	0.931	0.847	0.840	0.840	0.779	0.748
Epilepsy	0.986	0.964	0.971	0.957	0.957	0.949
Ering	0.919	0.874	0.133	0.852	0.904	0.874
EthanolConcentration	0.460	0.308	0.205	0.297	0.285	0.262
FaceDetection	0.541	0.501	0.513	0.536	0.544	0.534
FingerMovements	0.590	0.480	0.580	0.470	0.460	0.560
HandMovementDirection	0.432	0.338	0.351	0.324	0.243	0.243
Handwriting	0.428	0.515	0.451	0.249	0.498	0.225
Heartbeat	0.751	0.683	0.741	0.746	0.751	0.746
InsectWingbeat	0.449	0.466	0.156	0.469	0.264	0.105
JapaneseVowels	0.978	0.984	0.989	0.978	0.930	0.978
LSST	0.640	0.537	0.509	0.595	0.474	0.408
Libras	0.900	0.867	0.883	0.817	0.822	0.656
MotorImagery	0.520	0.510	0.580	0.500	0.610	0.500
NATOPS	0.972	0.928	0.917	0.911	0.822	0.850
PEMS-SF	0.884	0.682	0.676	0.699	0.734	0.740
PenDigits	0.987	0.989	0.981	0.979	0.974	0.560
PhonemeSpectra	0.292	0.233	0.222	0.207	0.252	0.085
RacketSports	0.908	0.855	0.855	0.776	0.816	0.809
SelfRegulationSCP1	0.860	0.812	0.843	0.799	0.823	0.754
SelfRegulationSCP2	0.600	0.578	0.539	0.550	0.533	0.550
SpokenArabicDigits	0.992	0.988	0.905	0.934	0.970	0.923
StandWalkJump	0.533	0.467	0.333	0.400	0.333	0.267
UWaveGestureLibrary	0.919	0.906	0.875	0.759	0.753	0.575

Table 5: Full results for **univariate** time-series forecasting using MSE and MAE as metrics. The reported scores for all comparison methods are taken from [32]. Note that for each method, pre-training and the downstream task are performed for each dataset individually.

Dataset	H	Ours		TS2Vec [32]		Informer [37]		LogTrans [21]		N-BEATS [28]		TCN [4]		LSTnet [20]	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh ₁	24	0.049	0.169	0.039	0.152	0.098	0.247	0.103	0.259	0.094	0.238	0.075	0.21	0.108	0.284
	48	0.079	0.215	0.062	0.191	0.158	0.319	0.167	0.328	0.21	0.367	0.227	0.402	0.175	0.424
	168	0.136	0.287	0.134	0.282	0.183	0.346	0.207	0.375	0.232	0.391	0.316	0.493	0.396	0.504
	336	0.158	0.315	0.154	0.31	0.222	0.387	0.23	0.398	0.232	0.388	0.306	0.495	0.468	0.593
	720	0.23	0.387	0.163	0.327	0.269	0.435	0.273	0.463	0.322	0.49	0.39	0.557	0.659	0.766
ETTh ₂	24	0.077	0.21	0.09	0.229	0.093	0.24	0.102	0.255	0.198	0.345	0.103	0.249	3.554	0.445
	48	0.102	0.247	0.124	0.273	0.155	0.314	0.169	0.348	0.234	0.386	0.142	0.29	3.19	0.474
	168	0.153	0.309	0.208	0.36	0.232	0.389	0.246	0.422	0.331	0.453	0.227	0.376	2.8	0.595
	336	0.183	0.339	0.213	0.369	0.263	0.417	0.267	0.437	0.431	0.508	0.296	0.43	2.753	0.738
	720	0.207	0.368	0.214	0.374	0.277	0.431	0.303	0.493	0.437	0.517	0.325	0.463	2.878	1.044
ETTh ₁	24	0.017	0.098	0.015	0.092	0.03	0.137	0.065	0.202	0.054	0.184	0.041	0.157	0.09	0.206
	48	0.031	0.132	0.027	0.126	0.069	0.203	0.078	0.22	0.19	0.361	0.101	0.257	0.179	0.306
	96	0.052	0.174	0.044	0.161	0.194	0.372	0.199	0.386	0.183	0.353	0.142	0.311	0.272	0.399
	288	0.109	0.257	0.103	0.246	0.401	0.554	0.411	0.572	0.186	0.362	0.318	0.472	0.462	0.558
	672	0.162	0.316	0.156	0.307	0.512	0.644	0.598	0.702	0.197	0.368	0.397	0.547	0.639	0.697
Electricity	24	0.272	0.386	0.26	0.288	0.251	0.275	0.528	0.447	0.427	0.33	0.263	0.279	0.281	0.287
	48	0.303	0.402	0.319	0.324	0.346	0.339	0.409	0.414	0.551	0.392	0.373	0.344	0.381	0.366
	168	0.337	0.431	0.427	0.394	0.544	0.424	0.959	0.612	0.893	0.538	0.609	0.462	0.599	0.5
	336	0.36	0.449	0.565	0.474	0.713	0.512	1.079	0.639	1.035	0.669	0.855	0.606	0.823	0.624
	720	0.359	0.453	0.861	0.643	1.182	0.806	1.001	0.714	1.548	0.881	1.263	0.858	1.278	0.906
Avg.		0.169	0.297	0.209	0.296	0.31	0.39	0.37	0.434	0.399	0.426	0.338	0.413	1.099	0.536

Table 6: Full results for **multivariate** time-series forecasting using MSE and MAE as metrics. The reported scores for all comparison methods are taken from [32]. Note that for each method, pre-training and the downstream task are performed for each dataset individually.

Dataset	H	Ours		TS2Vec [32]		Informer [37]		TCN [4]		LogTrans [21]		LSTnet [20]	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh ₁	24	0.504	0.518	0.599	0.534	0.577	0.549	0.767	0.612	0.686	0.604	1.293	0.901
	48	0.547	0.548	0.629	0.555	0.685	0.625	0.713	0.617	0.766	0.757	1.456	0.96
	168	0.689	0.636	0.755	0.636	0.931	0.752	0.995	0.738	1.002	0.846	1.997	1.214
	336	0.78	0.687	0.907	0.717	1.128	0.873	1.175	0.8	1.362	0.952	2.655	1.369
	720	0.813	0.69	1.048	0.79	1.215	0.896	1.453	1.311	1.397	1.291	2.143	1.38
ETTh ₂	24	0.293	0.409	0.398	0.461	0.72	0.665	1.365	0.888	0.828	0.75	2.742	1.457
	48	0.412	0.493	0.58	0.573	1.457	1.001	1.395	0.96	1.806	1.034	3.567	1.687
	168	0.86	0.737	1.901	1.065	3.489	1.515	3.166	1.407	4.07	1.681	3.242	2.513
	336	1.003	0.805	2.304	1.215	2.723	1.34	3.256	1.481	3.875	1.763	2.544	2.591
	720	1.013	0.803	2.65	1.373	3.467	1.473	3.69	1.588	3.913	1.552	4.625	3.709
ETTh ₁	24	0.342	0.41	0.443	0.436	0.323	0.369	0.324	0.374	0.419	0.412	1.968	1.17
	48	0.438	0.48	0.582	0.515	0.494	0.503	0.477	0.45	0.507	0.583	1.999	1.215
	96	0.482	0.511	0.622	0.549	0.678	0.614	0.636	0.602	0.768	0.792	2.762	1.542
	288	0.584	0.576	0.709	0.609	1.056	0.786	1.27	1.351	1.462	1.32	1.257	2.076
	672	0.685	0.634	0.786	0.655	1.192	0.926	1.381	1.467	1.669	1.461	1.917	2.941
Electricity	24	0.274	0.375	0.287	0.374	0.312	0.387	0.305	0.384	0.297	0.374	0.356	0.419
	48	0.288	0.384	0.307	0.388	0.392	0.431	0.317	0.392	0.316	0.389	0.429	0.456
	168	0.301	0.393	0.332	0.407	0.515	0.509	0.358	0.423	0.426	0.466	0.372	0.425
	336	0.305	0.398	0.349	0.42	0.759	0.625	0.349	0.416	0.365	0.417	0.352	0.409
	720	0.317	0.409	0.375	0.438	0.969	0.788	0.447	0.486	0.344	0.403	0.38	0.443
Avg.		0.546	0.545	0.828	0.635	1.154	0.781	1.192	0.837	1.314	0.892	1.903	1.444