HAL open science

Unsupervised Scalable Representation Learning for Multivariate Time Series

Jean-Yves Franceschi, Aymeric Dieuleveut, Martin Jaggi

To cite this version:

Jean-Yves Franceschi, Aymeric Dieuleveut, Martin Jaggi. Unsupervised Scalable Representation Learning for Multivariate Time Series. Thirty-third Conference on Neural Information Processing Systems, Neural Information Processing Systems Foundation, Dec 2019, Vancouver, Canada. hal-01998101v3

HAL Id: hal-01998101

https://hal.science/hal-01998101v3

Submitted on 1 Nov 2019 (v3), last revised 18 Dec 2019 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Unsupervised Scalable Representation Learning for Multivariate Time Series

Jean-Yves Franceschi*
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
jean-yves.franceschi@lip6.fr

Aymeric Dieuleveut
MLO, EPFL, Lausanne CH-1015, Switzerland
CMAP, Ecole Polytechnique, Palaiseau, France
aymeric.dieuleveut@polytechnique.edu

Martin Jaggi
MLO, EPFL, Lausanne CH-1015, Switzerland
martin.jaggi@epfl.ch

Abstract

Time series constitute a challenging data type for machine learning algorithms, due to their highly variable lengths and sparse labeling in practice. In this paper, we tackle this challenge by proposing an unsupervised method to learn universal embeddings of time series. Unlike previous works, it is scalable with respect to their length and we demonstrate the quality, transferability and practicability of the learned representations with thorough experiments and comparisons. To this end, we combine an encoder based on causal dilated convolutions with a novel triplet loss employing time-based negative sampling, obtaining general-purpose representations for variable length and multivariate time series.

1 Introduction

We investigate in this work the topic of unsupervised general-purpose representation learning for time series. In spite of the increasing amount of work about representation learning in fields like natural language processing (Young et al., 2018) or videos (Denton & Birodkar, 2017), few articles explicitly deal with general-purpose representation learning for time series without structural assumption on non-temporal data.

This problem is indeed challenging for various reasons. First, real-life time series are rarely or sparsely labeled. Therefore, *unsupervised* representation learning would be strongly preferred. Second, methods need to deliver compatible representations while allowing the input time series to have unequal lengths. Third, scalability and efficiency both at training and inference time is crucial, in the sense that the techniques must work for both short and long time series encountered in practice.

Hence, we propose in the following an *unsupervised* method to learn *general-purpose representations* for *multivariate* time series that comply with the issues of *varying and potentially high lengths* of the studied time series. To this end, we introduce a novel unsupervised loss training a scalable encoder, shaped as a deep convolutional neural network with dilated convolutions (Oord et al., 2016) and outputting fixed-length vector representations regardless of the length of its output. This loss is built

*Work partially done while studying at ENS de Lyon and MLO, EPFL.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

as a triplet loss employing time-based negative sampling, taking advantage of the encoder resilience to time series of unequal lengths. To our knowledge, it is the first fully unsupervised triplet loss in the literature of time series.

We assess the quality of the learned representations on various datasets to ensure their universality. In particular, we test how our representations can be used for classification tasks on the standard datasets in the time series literature, compiled in the UCR repository (Dau et al., 2018). We show that our representations are *general* and *transferable*, and that our method *outperforms concurrent unsupervised methods* and even *matches the state of the art* of non-ensemble supervised classification techniques. Moreover, since UCR time series are exclusively univariate and mostly short, we also evaluate our representations on the recent UEA multivariate time series repository (Bagnall et al., 2018), as well as on a real-life dataset including very long time series, on which we demonstrate *scalability*, *performance* and generalization ability *across different tasks* beyond classification.

This paper is organized as follows. Section 2 outlines previous works on unsupervised representation learning, triplet losses and deep architectures for time series in the literature. Section 3 describes the unsupervised training of the encoder, while Section 4 details the architecture of the latter. Finally, Section 5 provides results of the experiments that we conducted to evaluate our method.

2 Related Work

Unsupervised learning for time series. To our knowledge, apart from those dealing with videos (Srivastava et al., 2015; Denton & Birodkar, 2017; Villegas et al., 2017), few recent works tackle unsupervised representation learning for time series. Fortuin et al. (2019) deal with a related but different problem to this work, by learning temporal representations of time series that represent well their evolution. Hyvarinen & Morioka (2016) learn representations on evenly sized subdivisions of time series by learning to discriminate between those subdivisions from these representations. Lei et al. (2017) expose an unsupervised method designed so that the distances between learned representations mimic a standard distance (Dynamic Time Warping, DTW) between time series. Malhotra et al. (2017) design an encoder as a recurrent neural network, jointly trained with a decoder as a sequence-to-sequence model to reconstruct the input time series from its learned representation. Finally, Wu et al. (2018a) compute feature embeddings generated in the approximation of a carefully designed and efficient kernel.

However, these methods either are not scalable nor suited to long time series (due to the sequential nature of a recurrent network, or to the use of DTW with a quadratic complexity with respect to the input length), are tested on no or very few standard datasets and with no publicly available code, or do not provide sufficient comparison to assess the quality of the learned representations. Our scalable model and extensive analysis aim at overcoming these issues, besides outperforming these methods.

Triplet losses. Triplet losses have recently been widely used in various forms for representation learning in different domains (Mikolov et al., 2013; Schroff et al., 2015; Wu et al., 2018b) and have also been theoretically studied (Arora et al., 2019), but have not found much use for time series apart from audio (Bredin, 2017; Lu et al., 2017; Jansen et al., 2018), and never, to our knowledge, in a fully unsupervised setting, as existing works assume the existence of class labels or annotations in the training data. Closer to our work even though focusing on a different, more specific task, Turpault et al. (2019) learn audio embeddings in a semi-supervised setting, while partially relying on specific transformations of the training data to sample positive samples in the triplet loss; Logeswaran & Lee (2018) train a sentence encoder to recognize, among randomly chosen sentences, the true context of another sentence, which is a difficult method to adapt to time series. Our method instead relies on a more natural choice of positive samples, learning similarities using subsampling.

Convolutional networks for time series. Deep convolutional neural networks have recently been successfully applied to time series classification tasks (Cui et al., 2016; Wang et al., 2017), showing competitive performance. Dilated convolutions, popularized by WaveNet (Oord et al., 2016) for audio generation, have been used to improve their performance and were shown to perform well as sequence-to-sequence models for time series forecasting (Bai et al., 2018) using an architecture that inspired ours. These works particularly show that dilated convolutions help to build networks for sequential tasks outperforming recurrent neural networks in terms of both efficiency and prediction performance.

3 Unsupervised Training

We seek to train an encoder-only architecture, avoiding the need to jointly train with a decoder as in autoencoder-based standard representation learning methods as done by Malhotra et al. (2017), since those would induce a larger computational cost. To this end, we introduce a novel triplet loss for time series, inspired by the successful and by now classic word representation learning method known as word2vec (Mikolov et al., 2013). The proposed triplet loss uses original time-based sampling strategies to overcome the challenge of learning on unlabeled data. As far as we know, this work is the first in the time series literature to rely on a triplet loss in a fully unsupervised setting.

The objective is to ensure that similar time series obtain similar representations, with no supervision to learn such similarity. Triplet losses help to achieve the former (Schroff et al., 2015), but require to provide pairs of similar inputs, thus challenging the latter. While previous supervised works for time series using triplet losses assume that data is annotated, we introduce an unsupervised time-based criterion to select pairs of similar time series and taking into account time series of varying lengths, by following word2vec's intuition. The assumption made in the CBOW model of word2vec is twofold. The representation of the *context* of a word should probably be, on one hand, close to the one of this word (Goldberg & Levy, 2014), and, on the other hand, distant from the one of randomly chosen words, since they are probably unrelated to the original word's context. The corresponding loss then pushes pairs of (context, word) and (context, random word) to be linearly separable. This is called *negative sampling*.

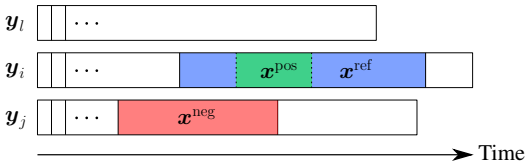


Figure 1: Choices of $\boldsymbol{x}^{\text{ref}}$, $\boldsymbol{x}^{\text{pos}}$ and $\boldsymbol{x}^{\text{neg}}$.

To adapt this principle to time series, we consider (see Figure 1 for an illustration) a random subseries[2] $\boldsymbol{x}^{\text{ref}}$ of a given time series $\boldsymbol{y}_i$. Then, on one hand, the representation of $\boldsymbol{x}^{\text{ref}}$ should be close to the one of any of its subseries $\boldsymbol{x}^{\text{pos}}$ (a *positive* example). On the other hand, if we consider another subseries $\boldsymbol{x}^{\text{neg}}$ (a *negative* example) chosen at random (in a different random time series $\boldsymbol{y}_j$ if several series are available, or in the same time series if it is long enough and not stationary), then its representation should be distant from the one of $\boldsymbol{x}^{\text{ref}}$. Following the analogy with word2vec, $\boldsymbol{x}^{\text{pos}}$ corresponds to a word, $\boldsymbol{x}^{\text{ref}}$ to its context, and $\boldsymbol{x}^{\text{neg}}$ to a random word. To improve the stability and convergence of the training procedure as well as the experimental results of our learned representations, we introduce, as in word2vec, several negative samples $(\boldsymbol{x}_k^{\text{neg}})_{k\in[\![1,K]\!]}$, chosen independently at random.

The objective corresponding to these choices to minimize during training can be thought of the one of word2vec with its shallow network replaced by a deep network $\boldsymbol{f}(\cdot,\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$, or formally

$$-\log\Big(\sigma\Big(\boldsymbol{f}\big(\boldsymbol{x}^{\text{ref}},\boldsymbol{\theta}\big)^\top \boldsymbol{f}(\boldsymbol{x}^{\text{pos}},\boldsymbol{\theta})\Big)\Big) - \sum_{k=1}^{K}\log\Big(\sigma\Big(-\boldsymbol{f}\big(\boldsymbol{x}^{\text{ref}},\boldsymbol{\theta}\big)^\top \boldsymbol{f}(\boldsymbol{x}_k^{\text{neg}},\boldsymbol{\theta})\Big)\Big), \tag{1}$$

where σ is the sigmoid function. This loss pushes the computed representations to distinguish between $\boldsymbol{x}^{\text{ref}}$ and $\boldsymbol{x}^{\text{neg}}$, and to assimilate $\boldsymbol{x}^{\text{ref}}$ and $\boldsymbol{x}^{\text{pos}}$. Overall, the training procedure consists in traveling through the training dataset for several epochs (possibly using mini-batches), picking tuples $\big(\boldsymbol{x}^{\text{ref}},\boldsymbol{x}^{\text{pos}},(\boldsymbol{x}_k^{\text{neg}})_k\big)$ at random as detailed in Algorithm 1, and performing a minimization step on the corresponding loss for each pair, until training ends. The overall computational and memory cost is $\mathcal{O}(K\cdot c(\boldsymbol{f}))$, where $c(\boldsymbol{f})$ is the cost of evaluating and backpropagating through $\boldsymbol{f}$ on a time series; thus this unsupervised training is scalable as long as the encoder architecture is scalable as well.

The length of the negative examples is chosen at random in Algorithm 1 for the most general case; however, their length can also be the same for all samples and equal to $\text{size}(\boldsymbol{x}^{\text{pos}})$. The latter case is suitable when all time series in the dataset have equal lengths, and speeds up the training procedure thanks to computation factorizations; the former case is only used when time series in the dataset do not have the same lengths, as we experimentally saw no other difference than time efficiency between the two cases. In our experiments, we do not cap the lengths of $\boldsymbol{x}^{\text{ref}}$, $\boldsymbol{x}^{\text{pos}}$ and $\boldsymbol{x}^{\text{neg}}$ since they are already limited by the length of the train time series, which corresponds to scales of lengths on which our representations are tested.

[2] I.e., a subsequence of a time series composed by consecutive time steps of this time series.

Algorithm 1: Choices of $\boldsymbol{x}^{\text{ref}}$, $\boldsymbol{x}^{\text{pos}}$ and $(\boldsymbol{x}_k^{\text{neg}})_{k \in [\![1,K]\!]}$ for an epoch over the set $(\boldsymbol{y}_i)_{i \in [\![1,N]\!]}$.

for $i \in [\![1, N]\!]$ **with** $s_i = \text{size}(\boldsymbol{y}_i)$ **do**
  pick $s^{\text{pos}} = \text{size}(\boldsymbol{x}^{\text{pos}})$ in $[\![1, s_i]\!]$ and $s^{\text{ref}} = \text{size}\big(\boldsymbol{x}^{\text{ref}}\big)$ in $[\![s^{\text{pos}}, s_i]\!]$ uniformly at random;
  pick $\boldsymbol{x}^{\text{ref}}$ uniformly at random among subseries of $\boldsymbol{y}_i$ of length s^{ref};
  pick $\boldsymbol{x}^{\text{pos}}$ uniformly at random among subseries of $\boldsymbol{x}^{\text{ref}}$ of length s^{pos};
  pick uniformly at random $i_k \in [\![1, N]\!]$, then $s_k^{\text{neg}} = \text{size}(\boldsymbol{x}_k^{\text{neg}})$ in $[\![1, \text{size}(\boldsymbol{y}_k)]\!]$ and finally $\boldsymbol{x}_k^{\text{neg}}$ among subseries of $\boldsymbol{y}_k$ of length s_k^{neg}, for $k \in [\![1, K]\!]$.

We highlight that this time-based triplet loss leverages the ability of the chosen encoder to take as input time series of different lengths. By training the encoder on a range of input lengths going from one to the length of the longest time series in the train set, it becomes able to output meaningful and transferable representations regardless of the input length, as shown in Section 5.

This training procedure is interesting in that it is efficient enough to be run over long time series (see Section 5) with a scalable encoder (see Section 4), thanks to its decoder-less design and the separability of the loss, on which a backpropagation per term can be performed to save memory.[3]

4 Encoder Architecture

We explain and present in this section our choice of architecture for the encoder, which is motivated by three requirements: it must extract relevant information from time series; it needs to be time- and memory-efficient, both for training and testing; and it has to allow variable-length inputs. We choose to use deep neural networks with *exponentially dilated causal convolutions* to handle time series. While they have been popularized in the context of sequence generation (Oord et al., 2016), they have never been used for unsupervised time series representation learning. They offer several advantages.

Compared to recurrent neural networks, which are inherently designed for sequence-modeling tasks and thus sequential, these networks are scalable as they allow efficient parallelization on modern hardware such as GPUs. Besides this demonstrated efficiency, exponentially dilated convolutions have also been introduced to better capture, compared to full convolutions, long-range dependencies at constant depth by exponentially increasing the receptive field of the network (Oord et al., 2016; Yu & Koltun, 2016; Bai et al., 2018).

Convolutional networks have also been demonstrated to be performant on various aspects for sequential data. For instance, recurrent networks are known to be subject to the issue of exploding and vanishing gradients, due to their recurrent nature (Goodfellow et al., 2016, Chapter 10.9). While significant work has been done to tackle this issue and improve their ability to capture long-term dependencies, such as the LSTM (Hochreiter & Schmidhuber, 1997), recurrent networks are still outperformed by convolutional networks on this aspect (Bai et al., 2018). On the specific domains of time series classification, which is an essential part of our experimental evaluation, and forecasting, deep neural networks have recently been successfully used (Bai et al., 2018; Ismail Fawaz et al., 2019).

Our model is particularly based on stacks of dilated *causal* convolutions (see Figure 2a), which map a sequence to a sequence of the same length, such that the i-th element of the output sequence is computed using only values up until the i-th element of the input sequence, for all i. It is thus called causal, since the output value corresponding to a given time step is not computed using future input values. Causal convolutions allow alleviating the disadvantage of not using recurrent networks at testing time. Indeed, recurrent networks can be used in an online fashion, thus saving memory and computation time during testing. In our case, causal convolutions organize the computational graph so that, in order to update its output when an element is added at the end of the input time series, one only has to evaluate the highlighted graph shown in Figure 2a rather than the full graph.

Inspired by Bai et al. (2018), we build each layer of our network as a combination of causal convolutions, weight normalizations (Salimans & Kingma, 2016), leaky ReLUs and residual connections (see Figure 2b). Each of these layers is given an exponentially increasing dilation parameter (2^i for the

[3] We used this optimization for multivariate or long (with length higher than 10 000) time series.

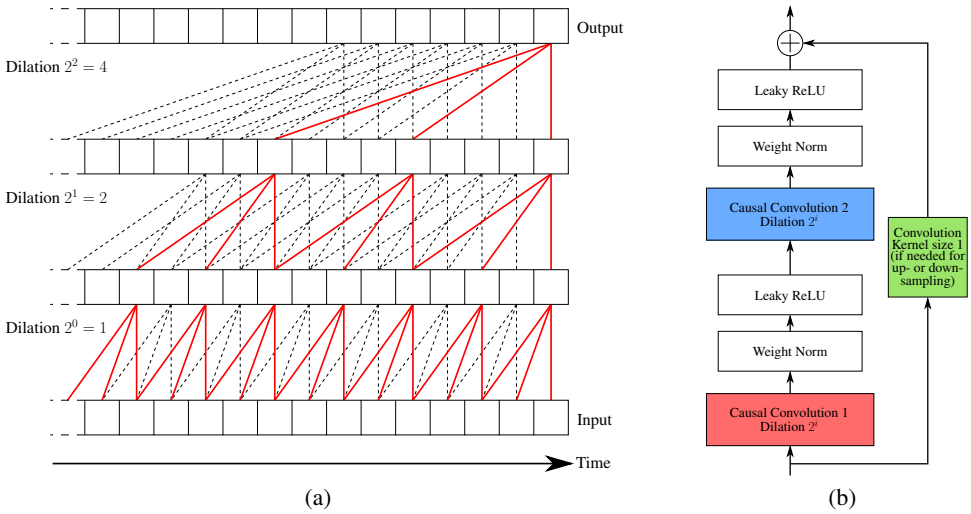


Figure 2: (a) Illustration of three stacked dilated causal convolutions. Lines between each sequence represent their computational graph. Red solid lines highlight the dependency graph for the computation of the last value of the output sequence, showing that no future value of the input time series is used to compute it. (b) Composition of the i-th layer of the chosen architecture.

i-th layer). The output of this causal network is then given to a global max pooling layer squeezing the temporal dimension and aggregating all temporal information in a fixed-size vector (as proposed by Wang et al. (2017) in a supervised setting with full convolutions). A linear transformation of this vector is then the output of the encoder, with a fixed, independent from the input length, size.

5 Experimental Results

We review in this section experiments conducted to investigate the relevance of the learned representations. The code corresponding to these experiments is attached in the supplementary material and is publicly available.[4] The full training process and hyperparameter choices are detailed in the supplementary material, Sections S1 and S2. We used Python 3 for implementation, with PyTorch 0.4.1 (Paszke et al., 2017) for neural networks and scikit-learn (Pedregosa et al., 2011) for SVMs. Each encoder was trained using the Adam optimizer (Kingma & Ba, 2015) on a single Nvidia Titan Xp GPU with CUDA 9.0, unless stated otherwise.

Selecting hyperparameters for an unsupervised method is challenging since the plurality of downstream tasks is usually supervised. Therefore, as Wu et al. (2018a), we choose for each considered dataset archive a single set of hyperparameters regardless of the downstream task. Moreover, we highlight that we perform *no hyperparameter optimization* of the unsupervised encoder architecture and training parameters for any task, unlike other unsupervised works such as TimeNet (Malhotra et al., 2017). Particularly, for classification tasks, *no label* was used during the encoder training.

5.1 Classification

We first assess the *quality* of our learned representations on supervised tasks in a standard manner (Xu et al., 2003; Dosovitskiy et al., 2014) by using them for time series classification. In this setting, we show that (1) our method outperforms state-of-the-art unsupervised methods, and notably achieves performance close to the supervised state of the art, (2) strongly outperforms supervised deep learning methods when data is only sparsely labeled, (3) produces transferable representations.

For each considered dataset with a train / test split, we unsupervisedly train an encoder using its train set. We then train an SVM with radial basis function kernel on top of the learned features using the train labels of the dataset, and output the corresponding classification score on the test set. As our

[4] https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries.

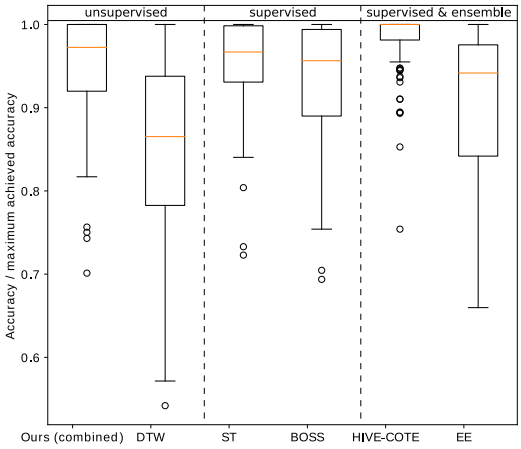


Figure 3: Boxplot of the ratio of the accuracy versus maximum achieved accuracy (higher is better) for compared methods on the first 85 UCR datasets.

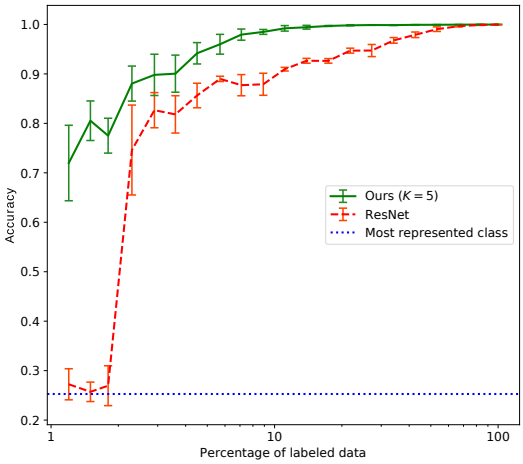


Figure 4: Accuracy of ResNet and our method with respect to the ratio of labeled data on TwoPatterns. Error bars correspond to the standard deviation over five runs per point for each method.

training procedure encourages representations of different time series to be separable, observing the classification performance of a simple SVM on these features allows to check their quality (Wu et al., 2018a). Using SVMs also allows, when the encoder is trained, an *efficient* training both in terms of time (training is a matter of minutes in most cases) and space.

As K has a significant impact on the performance, we present a *combined* version of our method, where representations computed by encoders trained with different values of K (see Section S2 for more details) are concatenated. This enables our learned representations with different parameters to complement each other, and to remove some noise in the classification scores.

5.1.1 Univariate Time Series

We present accuracy scores for all 128 datasets of the new iteration of the UCR archive (Dau et al., 2018), which is a standard set of varied univariate datasets. We report in Table 1 scores for only some UCR datasets, while scores for all datasets are reported in the supplementary material, Section S3.

We first compare our scores to the two concurrent methods of this work, TimeNet (Malhotra et al., 2017) and RWS (Wu et al., 2018a), which are two unsupervised methods also training a simple classifier on top of the learned representations, and reporting their results on a few UCR datasets. We also compare on the first 85 datasets of the archive[5] to the four best classifiers of the supervised state of the art studied by Bagnall et al. (2017): COTE – replaced by its improved version HIVE-COTE (Lines et al., 2018) –, ST (Bostrom & Bagnall, 2015), BOSS (Schäfer, 2015) and EE (Lines & Bagnall, 2015). HIVE-COTE is a powerful ensemble method using many classifiers in a hierarchical voting structure; EE is a simpler ensemble method; ST is based on shapelets and BOSS is a dictionary-based classifier.[6] We also add DTW (one-nearest-neighbor classifier with DTW as measure) as a baseline. HIVE-COTE includes ST, BOSS, EE and DTW in its ensemble, and is thus expected to outperform them. Additionally, we compare our method to the ResNet method of Wang et al. (2017), which is the best supervised neural network method studied in the review of Ismail Fawaz et al. (2019).

Performance. Comparison with the unsupervised state of the art (Section S3, Table S3 of the supplementary material), indicates that our method *consistently matches or outperforms both unsupervised methods* TimeNet and RWS (on 11 out of 12 and 10 out of 11 UCR datasets), showing its

[5]The new UCR archive includes 43 new datasets on which no reproducible results of state-of-the-art methods have been produced yet. Still, we provide complete results for our method on these datasets in the supplementary material, Section S3, Table S4, along with those of DTW, the only other method for which they were available.

[6]While ST and BOSS are also ensembles of classifiers, we chose not to qualify both of them as ensembles since their ensemble only includes variations of the same novel classification method.

Table 1: Accuracy scores of variants of our method compared with other supervised and unsupervised methods, on some UCR datasets. Results for the whole archive are available in the supplementary material, Section S3, Tables S1, S2 and S4. Bold and underlined scores respectively indicate the best and second-best (when there is no tie for first place) performing methods.

Dataset	Unsupervised					Supervised			
	Ours				DTW	ST	BOSS	Ensemble	
	$K = 5$	$K = 10$	Combined	FordA				HIVE-COTE	EE
DiatomSizeReduction	**0.993**	0.984	**0.993**	0.974	0.967	0.925	0.931	0.941	0.944
ECGFiveDays	**1**	**1**	**1**	**1**	**1**	0.984	**1**	**1**	0.82
FordB	0.781	0.793	<u>0.81</u>	0.798	0.62	0.807	0.711	**0.823**	0.662
Ham	0.657	**0.724**	<u>0.695</u>	0.533	0.467	0.686	0.667	0.667	0.571
Phoneme	0.249	0.276	0.289	0.196	0.228	<u>0.321</u>	0.265	**0.382**	0.305
SwedishLeaf	0.925	0.914	<u>0.931</u>	0.925	0.792	0.928	0.922	**0.954**	0.915

performance. Unlike our work, code and full results on the UCR archive are not provided for these methods, hence the incomplete results.

When comparing to the supervised non-neural-network state of the art, we observe (see Figures S2 and S3 in the supplementary material) that our method is globally the second-best one (with average rank 2.92), only beaten by HIVE-COTE (1.71) and equivalent to ST (2.95). Thus, our unsupervised method beats several recognized supervised classifier, and is only preceded by a powerful ensemble method, which was expected since the latter takes advantage of numerous classifiers and data representations. Additionally, Figure 3 shows that our method has the second-best median for the ratio of accuracy over maximum achieved accuracy, behind HIVE-COTE and above ST. Finally, results reported from the study of Ismail Fawaz et al. (2019) for the fully supervised ResNet (Section S3, Table S3 of the supplementary material) show that it expectedly outperforms our method on 63% out of 71 UCR datasets.[7] Overall, our method achieves remarkable performance as it *is close to the best supervised neural network*, *matches the second-best studied non-neural-network supervised method*, and, in particular, is *at the level of the best performing method included in HIVE-COTE*.[8]

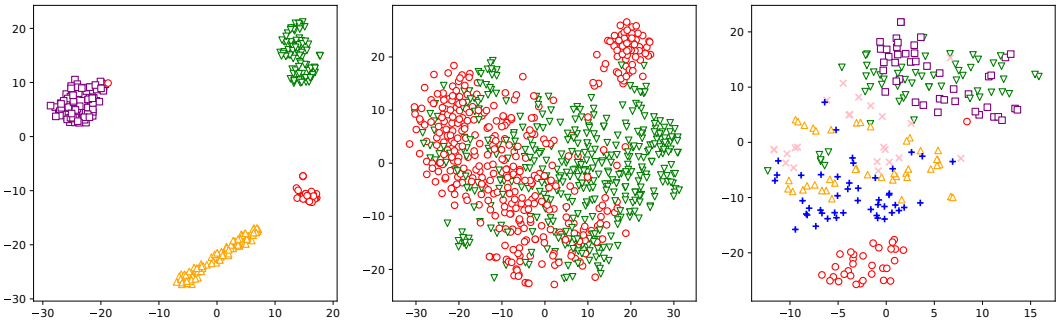
Sparse labeling. Taking advantage of their unsupervised training, we show that our representations can be successfully used on sparsely labeled datasets compared to supervised methods, since only the SVM is restricted to be learned on the small portion of labeled data. Figure 4 shows that an SVM trained on our representations of a randomly chosen labeled set *consistently outperforms the supervised neural network ResNet* trained on a labeled set of the same size, especially when the percentage of labeled data is small. For example, with only 1.5% of labeled data, we achieve an accuracy of 81%, against only 26% for ResNet, equivalent to a random classifier. Moreover, we exceed 99% of accuracy starting from 11% of labeled data, while ResNet only achieves this level of accuracy with more than 50% of labeled data. This shows the relevance of our method in semi-supervised settings, compared to fully supervised methods.

Representations metric space. Besides being suitable for classification purposes, the learned representations may also be used to define a meaningful measure between time series. Indeed, we train, instead of an SVM, a one-nearest-neighbor classifier with respect to the ℓ_2 distance on the same representations, and compare it to DTW, which uses the same classifier on the raw time series. As shown in S3, this version of our method outperforms DTW on 66% of the UCR datasets, showing the advantage of the learned representations even in a non-parametric classification setting. We also include quantitative experiments to assess the usefulness of comparing time series using the ℓ_2 distance between their representations with dimensionality reduction (Figure 5) and clustering (Section 5.2 and Figure 6) visualizations.

Transferability. We include in the comparisons the classification accuracy for each dataset of an SVM trained on this dataset using the representations computed by an encoder, which was trained *on another dataset* (FordA, with $K = 5$), to test the transferability of our representations. We observe

[7] Those results are incomplete as Ismail Fawaz et al. (2019) performed their experiments on the old version of the archive, whereas ours are performed on its most recent release where some datasets were changed.

[8] Our method could be included in HIVE-COTE, which could improve its performance, but this is beyond the scope of this work and requires technical work, as HIVE-COTE is implemented in Java and ours in Python.



(a) DiatomSizeReduction. (b) FordB. (c) OSULeaf.

Figure 5: Two-dimensional t-SNE (Maaten & Hinton, 2008) with perplexity 30 of the learned representations of three UCR test sets. Elements classes are distinguishable using their respective marker shapes and colors.

that the scores achieved by this SVM trained on transferred representations are close to the scores reported when the encoder is trained on the same dataset as the SVM, showing the *transferability* of our representations from a dataset to another, and from time series to other time series *with different lengths*. More generally, this transferability and the performance of simple classifiers on the representations we learn indicate that they are *universal* and *easy to make use of*.

5.1.2 Multivariate Time Series

To complement our evaluation on the UCR archive which exclusively contains univariate series, we evaluate our method on multivariate time series. This can be done by simply changing the number of input filters of the first convolutional layer of the proposed encoder. We test our method on all 30 datasets of the newly released UEA archive (Bagnall et al., 2018). Full accuracy scores are presented in the supplementary material, Section S4, Table S5.

The UEA archive has been designed as a first attempt to provide a standard archive for multivariate time series classification such as the UCR one for univariate series. As it has only been released recently, we could not compare our method to state-of-the-art classifiers for multivariate time series. However, we provide a comparison with DTW_D as baseline using results provided by Bagnall et al. (2018). DTW_D (dimension-Dependent DTW) is a possible extension of DTW in the multivariate setting, and is the best baseline studied by Bagnall et al. (2018). Overall, our method matches or outperforms DTW_D on 69% of the UEA datasets, which indicates good performance. As this archive is destined to grow and evolve in the future, and without further comparisons, no additional conclusion can be drawn.

5.2 Evaluation on Long Time Series

We show the *applicability* and *scalability* of our method on *long* time series without labeling for regression tasks, which could correspond to an industrial application and complements the performed tests on the UCR and UEA archives, whose datasets mostly contain short time series.

The Individual Household Electric Power Consumption (IHEPC) dataset from the UCI Machine Learning Repository (Dheeru & Karra Taniskidou, 2017) is a single time series of length $2\,075\,259$ monitoring the minute-averaged electricity consumption of one French household for four years. We split this time series into train (first 5×10^5 measurements, approximately a year) and test (remaining measurements). The encoder is trained over the train time series on a single Nvidia Tesla P100 GPU in no more than a few hours, showing that our training procedure is *scalable* to long time series.

We consider the learned encoder on two regression tasks involving two different input scales. We compute, for each time step of the time series, the representations of the last window corresponding to a day ($1\,440$ measurements) and a quarter ($12 \cdot 7 \cdot 1\,440$ measurements) *using the same encoder*. An example of application of the day-long representations is shown in Figure 6. The considered tasks consist in, for each time step, predicting the discrepancy between the mean value of the series

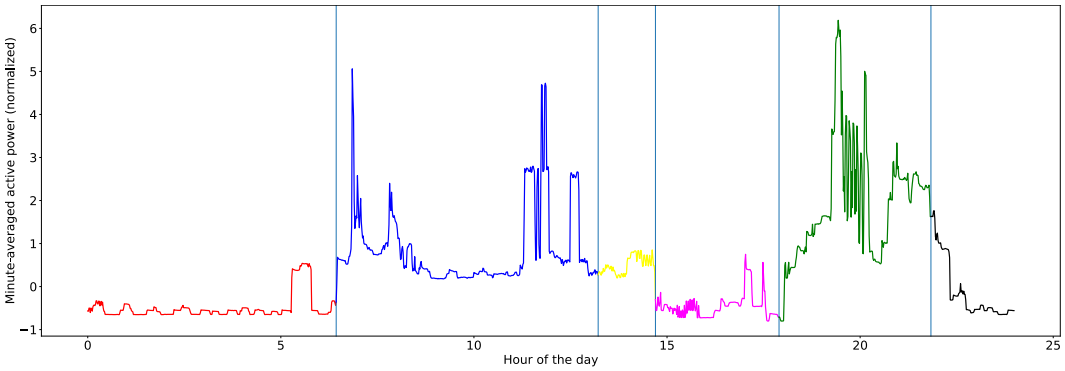


Figure 6: Minute-averaged electricity consumption for a single day, with respect to the hour of the day. Vertical lines and colors divide the day into six clusters, obtained with k-means clustering based on representations computed on a day-long sliding window. The clustering divides the day in meaningful portions (night, morning, afternoon, evening).

for the next period (either a day or quarter) and the one for the previous period. We compare linear regressors, trained using gradient descent, to minimize the mean squared error between the prediction and the target, applied either on the raw time series or the previously computed representations.

Results and execution times on an Nvidia Titan Xp GPU are presented in Table 2.[9] On *both scales of inputs*, our representations induce only a slightly degraded performance but provide a *large efficiency improvement*, due to their small size compared to the raw time series. This shows that a single encoder trained to minimize our time-based loss is able to output representations for different scales of input lengths that are also helpful for other tasks than classification, corroborating their *universality*.

Table 2: Results obtained on the IHEPC dataset.

Task	Metric	Representations	Raw values
Day	Test MSE	$\mathbf{8.92 \times 10^{-2}}$	$\mathbf{8.92 \times 10^{-2}}$
	Wall time	**12s**	3min 1s
Quarter	Test MSE	7.26×10^{-2}	$\mathbf{6.26 \times 10^{-2}}$
	Wall time	**9s**	1h 40min 15s

6 Conclusion

We present an unsupervised representation learning method for time series that is scalable and produces high-quality and easy-to-use embeddings. They are generated by an encoder formed by dilated convolutions that admits variable-length inputs, and trained with an efficient triplet loss using novel time-based negative sampling for time series. Conducted experiments show that these representations are universal and can easily and efficiently be used for diverse tasks such as classification, for which we achieve state-of-the-art performance, and regression.

Acknowledgements

We would like to acknowledge Patrick Gallinari, Sylvain Lamprier, Mehdi Lamrayah, Etienne Simon, Valentin Guiguet, Clara Gainon de Forsan de Gabriac, Eloi Zablocki, Antoine Saporta, Edouard Delasalles, Sidak Pal Singh, Andreas Hug, Jean-Baptiste Cordonnier, Andreas Loukas and François Fleuret for helpful comments and discussions at various stages of this project, as well as all contributors to the datasets and archives we used for this project (Dau et al., 2018; Bagnall et al., 2018; Dheeru & Karra Taniskidou, 2017).

We acknowledge financial support from the SFA-AM ETH Board initiative, the ANR project LOCUST (ANR-15-CE23-0027) and CLEAR (Center for LEArning and data Retrieval, joint laboratory with Thales[10]).

[9] While acting on representations of the same size, the quarterly linear regressor is slightly faster than the daily one because the number of quarters in the considered time series is smaller than the number of days.

[10] https://www.thalesgroup.com.

References

Arora, S., Khandeparkar, H., Khodak, M., Plevrakis, O., and Saunshi, N. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, May 2017.

Bagnall, A., Dau, H. A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., and Keogh, E. The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Bostrom, A. and Bagnall, A. Binary shapelet transform for multiclass time series classification. In *Big Data Analytics and Knowledge Discovery*, pp. 257–269, Cham, 2015. Springer International Publishing.

Bredin, H. Tristounet: Triplet loss for speaker turn embedding. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5430–5434, March 2017.

Cui, Z., Chen, W., and Chen, Y. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

Dau, H. A., Keogh, E., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G. The UCR time series classification archive, October 2018.

Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., and Zupan, B. Orange: Data mining toolbox in Python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

Denton, E. L. and Birodkar, V. Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems 30*, pp. 4414–4423. Curran Associates, Inc., 2017.

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017.

Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems 27*, pp. 766–774. Curran Associates, Inc., 2014.

Fortuin, V., Hüser, M., Locatello, F., Strathmann, H., and Rätsch, G. SOM-VAE: Interpretable discrete representation learning on time series. In *International Conference on Learning Representations*, 2019.

Goldberg, Y. and Levy, O. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

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

Hyvarinen, A. and Morioka, H. Unsupervised feature extraction by time-contrastive learning and nonlinear ICA. In *Advances in Neural Information Processing Systems 29*, pp. 3765–3773. Curran Associates, Inc., 2016.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, March 2019.

Jansen, A., Plakal, M., Pandya, R., Ellis, D. P. W., Hershey, S., Liu, J., Moore, R. C., and Saurous, R. A. Unsupervised learning of semantic audio representations. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 126–130, April 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Lei, Q., Yi, J., Vaculin, R., Wu, L., and Dhillon, I. S. Similarity preserving representation learning for time series analysis. *arXiv preprint arXiv:1702.03584*, 2017.

Lines, J. and Bagnall, A. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592, May 2015.

Lines, J., Taylor, S., and Bagnall, A. Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data*, 12(5):52:1–52:35, July 2018.

Logeswaran, L. and Lee, H. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*, 2018.

Lu, R., Wu, K., Duan, Z., and Zhang, C. Deep ranking: Triplet MatchNet for music metric learning. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 121–125, March 2017.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, November 2008.

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

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc., 2013.

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

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, October 2011.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems 29*, pp. 901–909. Curran Associates, Inc., 2016.

Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, November 2015.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, June 2015.

Srivastava, N., Mansimov, E., and Salakhudinov, R. Unsupervised learning of video representations using LSTMs. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 843–852, Lille, France, July 2015. PMLR.

Turpault, N., Serizel, R., and Vincent, E. Semi-supervised triplet loss based learning of ambient audio embeddings. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 760–764, May 2019.

Villegas, R., Yang, J., Hong, S., Lin, X., and Lee, H. Decomposing motion and content for natural video sequence prediction. *International Conference on Learning Representations*, 2017.

Wang, Z., Yan, W., and Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1585, May 2017.

Wu, L., Yen, I. E.-H., Yi, J., Xu, F., Lei, Q., and Witbrock, M. Random Warping Series: A random features method for time-series embedding. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 793–802. PMLR, April 2018a.

Wu, L. Y., Fisch, A., Chopra, S., Adams, K., Bordes, A., and Weston, J. Starspace: Embed all the things! In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b.

Xu, W., Liu, X., and Gong, Y. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pp. 267–273, New York, NY, USA, 2003. ACM.

Young, T., Hazarika, D., Poria, S., and Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75, August 2018.

Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.

Appendices

In these appendices, we provide our detailed training procedure for classification tasks, choices of hyperparameters, as well as the full experimental results of our method, compared to those of concurrent methods. Section S1 explains and discusses the exact training process for classification tasks. Section S2 details the choices of hyperparameters in all presented experiments. Section S3 reports accuracy scores of all variants of our method on the whole UCR archive (Dau et al., 2018), as well as comparisons with concurrent methods, when available. Section S4 provides accuracy scores for our method on the whole UEA archive (Bagnall et al., 2018). Finally, Section S5 discusses the importance of the choice of encoder by providing results obtained with our training procedure using an LSTM as encoder.

S1 Training Details

S1.1 Input Preprocessing

We preprocess datasets of the UCR archive that were not already normalized, as well as the IHEPC dataset, so that the set of time series values for each dataset has zero mean and unit variance. For each UEA dataset, each dimension of the time series was preprocessed independently from the other dimensions by normalizing in the same way its mean and variance.

S1.2 SVM Training

In order to train an SVM on the computed representations of the elements of the train set, we perform a hyperparameter optimization for the penalty C of the error term of the SVM by cross-validating it over the representations of the train set, thus only using the train labels. Note that if the train set or the number of training samples per class are too small, we choose a penalty $C = \infty$ for the SVM (which corresponds to no regularization).

S1.3 Behavior of the Learned Representations through Training

Classification accuracy evolution during training. As shown in Figure S1, our unsupervised training clearly makes the classification accuracy of the trained SVM increase with the number of optimization steps.

Numerical stability. The *Risk* R is defined as the expectation (taken over the random selection of the sequences $\left(\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{x}^{\mathrm{pos}}, \boldsymbol{x}^{\mathrm{neg}}\right)$) of the loss defined in Equation (1). This risk may decrease if all the representations $\boldsymbol{f}(\cdot, \boldsymbol{\theta})$ are scaled by a positive large number. For example, if for some $\boldsymbol{\theta}_0$, for (almost surely) any sequences $\left(\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{x}^{\mathrm{pos}}, \boldsymbol{x}^{\mathrm{neg}}\right)$, $\boldsymbol{f}\left(\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{\theta}_0\right)^\top \boldsymbol{f}(\boldsymbol{x}^{\mathrm{pos}}, \boldsymbol{\theta}_0) \geq 0$ and $\boldsymbol{f}\left(\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{\theta}_0\right)^\top \boldsymbol{f}(\boldsymbol{x}^{\mathrm{neg}}, \boldsymbol{\theta}_0) \leq 0$, then

$$R(\lambda, \boldsymbol{\theta}_0) := \mathbb{E}_{\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{x}^{\mathrm{pos}}, \boldsymbol{x}^{\mathrm{neg}}} \Big[-\log\Big(\sigma\Big(\lambda^2 \boldsymbol{f}\left(\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{\theta}_0\right)^\top \boldsymbol{f}(\boldsymbol{x}^{\mathrm{pos}}, \boldsymbol{\theta}_0)\Big)\Big) - \log\Big(\sigma\Big(-\lambda^2 \boldsymbol{f}\left(\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{\theta}_0\right)^\top \boldsymbol{f}(\boldsymbol{x}^{\mathrm{neg}}, \boldsymbol{\theta}_0)\Big)\Big)\Big] \quad (2)$$

is a decreasing function of λ, thus λ could diverge to infinity in order to minimize the loss. In other words, the parameters in $\boldsymbol{\theta}_0$ corresponding to the last linear layer could be linearly scaled up, and representations would "explode" (their norm would always increase through training). Such a phenomenon is not observed in practice, as the mean representation Euclidean norm lies around 20. There are two possible explanations for that: either the condition above is not satisfied (more generally, the loss is not reduced by increasing the representations) or the use of the sigmoid function, that has vanishing gradients, results in an increase of the representations that is too slow to be observed, or negligible with respect to other weight updates during optimization.

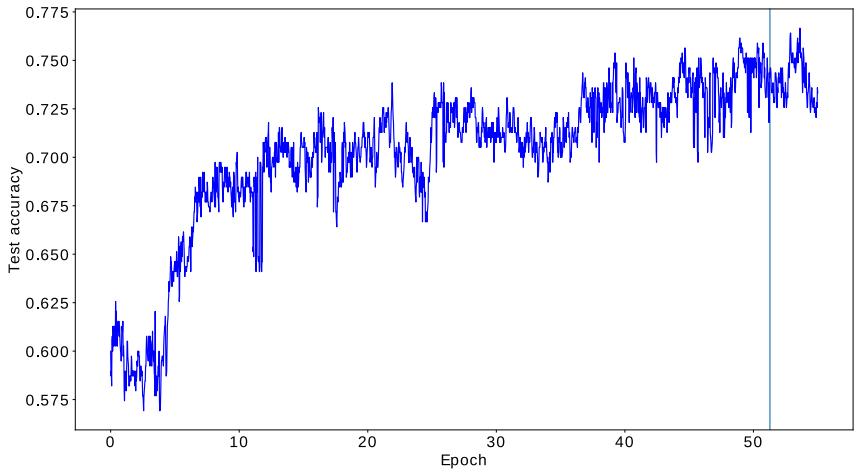


Figure S1: Evolution of the test accuracy during the training of the encoder on the CricketX dataset from the UCR archive (with $K = 10$), with respect to the number of completed epochs. The test labels were only used for monitoring purposes and the test accuracy was computed after each mini-batch optimization. The vertical line marks the epoch at which 2000 optimization steps were performed, at which point training is stopped in our tests. Test accuracy clearly increases during training.

S2 Hyperparameters

S2.1 Influence of K

As mentioned in Section 5, K can have a significant impact on the performance of the encoder. We notably observed that $K = 1$ leads to statistically significantly lower scores compared to scores obtained when trained with $K > 1$ on the UCR datasets, justifying the use of several negative examples during training. We did not observe any clear statistical difference between other values of K on the whole archive; however, we noticed important differences between different values of K when studying individual datasets. Therefore, we chose to combine several encoders trained with different values of K in order to avoid selecting it as a fixed hyperparameter.

S2.2 Detailed Choices of Hyperparameters

We train our models with the following parameters for time series classification. Note that *no hyperparameter optimization* was performed on the encoder hyperparameters.

- Optimizer: Adam (Kingma & Ba, 2015) with learning rate $\alpha = 0.001$ and decay rates $\beta = (0.9, 0.999)$.
- SVM: penalty $C \in \left\{10^i \mid i \in [\![-4, 4]\!]\right\} \cup \{\infty\}$.
- Encoder training:
  - number of negative samples: $K \in \{1, 2, 5, 10\}$ for univariate time series, $K \in \{5, 10, 20\}$ for multivariate ones;
  - batch size: 10;
  - number of optimizations steps: 2000 for $K \geq 10$ (i.e., 20 epochs for a dataset of size 1000), 1500 otherwise.
- Architecture:
  - number of channels in the intermediary layers of the causal network: 40;
  - number of layers (depth of the causal network): 10;
  - kernel size of all convolutions: 3;
  - negative slope of the leaky ReLU activation: 0.01;
  - number of output channels of the causal network (before max pooling): 320;
  - dimension of the representations: 160.

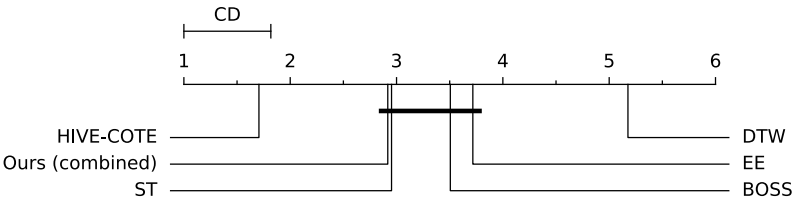


Figure S2: Critical difference diagram of the average ranks of the compared classifiers for the Nemenyi test, obtained with Orange (Demšar et al., 2013).

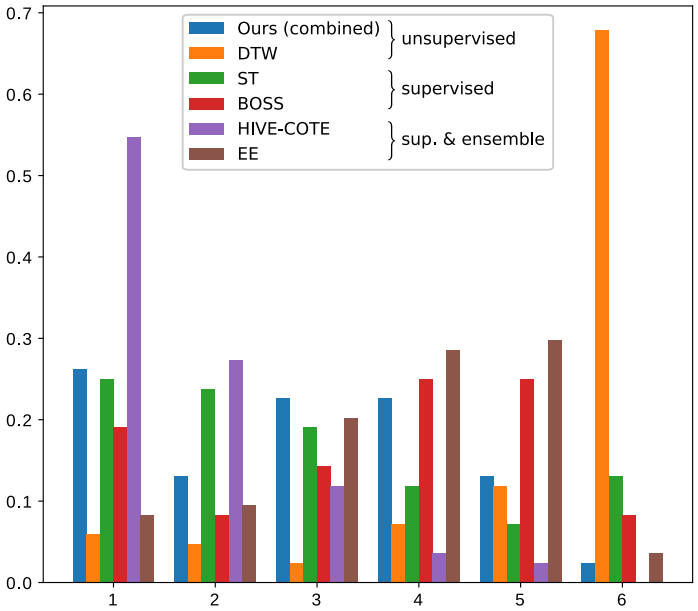


Figure S3: Distribution of ranks of compared methods for the first 85 UCR datasets.

For the Individual Household Electric Power Consumption dataset, changes are the following:

- number of negative samples: $K = 10$;
- batch size: 1;
- number of optimization steps: 400;
- number of channels in the intermediary layers of the causal network: 30;
- number of output channels of the causal network (before max pooling): 160;
- dimension of the representations: 80.

S3 Univariate Time Series

Full results corresponding to the first 85 UCR datasets for our method are presented in Table S1, while comparisons with DTW, ST, BOSS, HIVE-COTE and EE are shown in Figures S2 and S3 and Table S2,[S1] and comparisons with ResNet,[S2] TimeNet and RWS are shown in Table S3. Table S4 compiles the results of our method and of DTW[S3] for the newest 43 UCR datasets (except DodgerLoopDay, DodgerLoopGame and DodgerLoopWeekend which contain missing values).

[S1] Scores taken from http://www.timeseriesclassification.com/singleTrainTest.csv.
[S2] Scores taken from https://github.com/hfawaz/dl-4-tsc/blob/master/results/results-uea.csv (first iteration).
[S3] Scores taken from https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

Standard deviation. All UCR datasets are provided with a unique train / test split that we used in our experiments. Compared techniques (DTW, ST, BOSS, HIVE-COTE and EE) were also tested on 100 random train / test splits of these datasets by Bagnall et al. (2017) to produce a strong state-of-the-art evaluation, but we did not perform similar resamples as this is beyond the scope of this work and would require much more computations. Note that the scores for these methods used in this article are the ones corresponding to the original train / test split of the datasets.

As our method is based on random sampling, the reported scores may vary depending on the random seed. While we do not report standard deviation, the large number of tested datasets prevents large statistical error in the global evaluation of our method. The order of magnitude of accuracy variation between different runs of the combined version of our method is below 0.01 (for instance, on four different runs, the corresponding standard variations for, respectively, datasets DiatomSizeReduction, CricketX and UWaveGestureLibraryX are 0.0056, 0.0091 and 0.0053).

Table S1: Accuracy scores of variants of our method on the first 85 UCR datasets. "Combined (1-NN)" corresponds to learning a one-nearest-neighbor classifier, instead of an SVM, on the combined representations. Bold scores indicate the best performing method.

Dataset	Unsupervised						
	Ours						
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	Combined	Combined (1-NN)	FordA
Adiac	0.734	0.711	0.703	0.675	0.716	0.645	**0.76**
ArrowHead	**0.869**	0.829	0.754	0.766	0.829	0.817	0.817
Beef	**0.733**	0.567	0.7	0.667	0.7	0.6	0.667
BeetleFly	**0.9**	0.8	**0.9**	0.8	**0.9**	0.8	0.8
BirdChicken	0.7	0.8	**0.9**	0.85	0.8	0.75	**0.9**
Car	0.75	0.767	0.633	0.833	0.817	0.8	**0.85**
CBF	0.982	0.991	0.99	0.983	**0.994**	0.978	0.988
ChlorineConcentration	0.719	0.747	0.739	0.749	**0.782**	0.588	0.688
CinCECGTorso	0.702	**0.747**	0.682	0.713	0.74	0.693	0.638
Coffee	0.964	**1**	**1**	**1**	**1**	**1**	**1**
Computers	**0.688**	0.644	0.676	0.664	0.628	0.604	0.648
CricketX	0.736	0.71	0.7	0.713	**0.777**	0.741	0.682
CricketY	0.682	0.664	0.695	0.728	**0.767**	0.664	0.667
CricketZ	0.721	0.71	0.726	0.708	**0.764**	0.723	0.656
DiatomSizeReduction	0.99	0.987	**0.993**	0.984	**0.993**	0.967	0.974
DistalPhalanxOutlineCorrect	0.761	0.746	**0.775**	**0.775**	0.768	0.757	0.764
DistalPhalanxOutlineAgeGroup	0.719	**0.748**	0.719	0.727	0.734	0.683	0.727
DistalPhalanxTW	**0.698**	0.676	0.662	0.676	0.676	0.669	0.669
Earthquakes	**0.748**	**0.748**	**0.748**	**0.748**	**0.748**	0.64	**0.748**
ECG200	0.87	0.9	0.86	**0.94**	0.9	0.85	0.83
ECG5000	0.939	0.939	0.937	0.933	0.936	0.925	**0.94**
ECGFiveDays	**1**	**1**	**1**	**1**	**1**	0.999	**1**
ElectricDevices	0.709	0.7	0.712	0.707	**0.732**	0.646	0.676
FaceAll	0.764	**0.81**	0.733	0.786	0.802	0.75	0.734
FaceFour	0.807	0.864	0.795	**0.92**	0.875	0.864	0.83
FacesUCR	0.885	0.871	0.886	0.884	**0.918**	0.86	0.835
FiftyWords	0.763	0.734	0.727	0.732	**0.78**	0.716	0.745
Fish	0.903	0.909	0.891	0.891	0.88	0.823	**0.96**
FordA	0.923	0.922	0.927	0.928	**0.935**	0.863	0.927
FordB	0.786	0.788	0.781	0.793	**0.81**	0.748	0.798
GunPoint	0.953	0.987	0.987	0.98	**0.993**	0.833	0.987
Ham	0.648	0.686	0.657	**0.724**	0.695	0.533	0.533
HandOutlines	**0.922**	0.919	0.908	**0.922**	**0.922**	0.832	0.919
Haptics	0.445	0.435	0.432	**0.49**	0.455	0.354	0.474
Herring	**0.609**	0.594	0.578	0.594	0.578	0.563	0.578
InlineSkate	0.425	0.429	0.427	0.371	**0.447**	0.4	0.444
InsectWingbeatSound	0.61	0.592	0.617	0.597	**0.623**	0.506	0.599
ItalyPowerDemand	0.94	0.927	0.928	**0.954**	0.925	0.942	0.929
LargeKitchenAppliances	0.797	0.827	0.843	0.789	**0.848**	0.757	0.765
Lightning2	0.869	0.836	0.852	0.869	**0.918**	0.885	0.787
Lightning7	0.795	**0.822**	**0.822**	0.795	0.795	0.795	0.74
Mallat	0.962	0.931	0.947	0.951	**0.964**	0.944	0.916
Meat	0.917	0.867	0.867	**0.95**	**0.95**	0.9	0.867
MedicalImages	0.738	0.768	0.77	0.75	**0.784**	0.693	0.725
MiddlePhalanxOutlineCorrect	0.749	0.818	0.777	**0.825**	0.814	0.722	0.787
MiddlePhalanxOutlineAgeGroup	0.617	**0.662**	0.656	0.656	0.656	0.506	0.623
MiddlePhalanxTW	0.604	**0.61**	**0.61**	0.591	**0.61**	0.513	0.584
MoteStrain	**0.875**	0.854	0.867	0.851	0.871	0.853	0.823
NonInvasiveFetalECGThorax1	0.912	0.911	0.904	0.878	0.91	0.798	**0.925**
NonInvasiveFetalECGThorax2	0.925	0.925	0.918	0.919	0.927	0.82	**0.93**
OliveOil	0.867	0.833	0.867	0.867	**0.9**	0.833	**0.9**
OSULeaf	0.719	0.694	0.793	0.76	**0.831**	0.636	0.736
PhalangesOutlinesCorrect	**0.807**	0.796	0.795	0.784	0.801	0.752	0.784
Phoneme	0.264	0.265	0.249	0.276	**0.289**	0.197	0.196
Plane	0.99	**1**	0.99	0.99	0.99	**1**	0.981
ProximalPhalanxOutlineCorrect	**0.869**	0.863	0.856	0.859	0.859	0.801	**0.869**
ProximalPhalanxOutlineAgeGroup	0.849	**0.859**	0.844	0.844	0.854	0.805	0.839
ProximalPhalanxTW	**0.824**	0.815	0.761	0.771	**0.824**	0.717	0.785
RefrigerationDevices	0.531	0.507	0.547	0.515	0.517	0.475	**0.555**
ScreenType	0.408	0.411	**0.427**	0.416	0.413	0.389	0.384
ShapeletSim	**0.894**	0.5	0.628	0.672	0.817	0.772	0.517
ShapesAll	0.847	0.84	0.857	0.848	**0.875**	0.823	0.837
SmallKitchenAppliances	0.68	0.667	0.715	0.677	0.715	0.619	**0.731**
SonyAIBORobotSurface1	**0.93**	0.89	0.85	0.902	0.897	0.825	0.84
SonyAIBORobotSurface2	0.885	0.933	0.928	0.889	**0.934**	0.885	0.832
StarLightCurves	0.96	0.966	0.958	0.964	0.965	0.893	**0.968**
Strawberry	0.951	0.946	**0.954**	**0.954**	0.946	0.903	0.946
SwedishLeaf	0.907	0.925	0.925	0.914	**0.931**	0.891	0.925
Symbols	0.937	0.931	**0.965**	0.963	**0.965**	0.933	0.945
SyntheticControl	0.98	0.983	**0.987**	**0.987**	0.983	0.977	0.977

Table S1: Accuracy scores of variants of our method on the first 85 UCR datasets. "Combined (1-NN)" corresponds to learning a one-nearest-neighbor classifier, instead of an SVM, on the combined representations. Bold scores indicate the best performing method.

Dataset	Unsupervised						
	Ours						
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	Combined	Combined (1-NN)	FordA
ToeSegmentation1	0.868	**0.961**	0.93	0.939	0.952	0.851	0.899
ToeSegmentation2	0.869	0.892	0.838	**0.9**	0.885	**0.9**	**0.9**
Trace	**1**	**1**	**1**	0.99	**1**	**1**	**1**
TwoLeadECG	0.996	0.991	0.996	**0.999**	0.997	0.988	0.993
TwoPatterns	0.998	**1**	**1**	0.999	**1**	0.998	0.992
UWaveGestureLibraryX	0.795	0.791	0.806	0.785	**0.811**	0.762	0.784
UWaveGestureLibraryY	0.716	0.717	0.702	0.71	**0.735**	0.666	0.697
UWaveGestureLibraryZ	0.738	0.735	0.741	0.757	**0.759**	0.679	0.729
UWaveGestureLibraryAll	0.893	0.887	0.903	0.896	**0.941**	0.838	0.865
Wafer	0.991	**0.995**	0.993	0.992	0.993	0.987	**0.995**
Wine	0.704	0.815	0.852	0.815	**0.87**	0.5	0.685
WordSynonyms	0.63	0.646	0.676	0.691	**0.704**	0.633	0.641
Worms	0.662	**0.74**	0.688	0.727	0.714	0.597	0.688
WormsTwoClass	0.753	0.766	0.74	0.792	**0.818**	0.805	0.753
Yoga	0.824	0.854	0.831	0.837	**0.878**	0.837	0.828

Table S2: Accuracy scores of the combined version of our method compared with those of DTW (unsupervised), ST and BOSS (supervised) and HIVE-COTE and EE (supervised ensemble methods), on the first 85 UCR datasets (results on the full archive were not available for comparisons). Bold scores indicate the best performing method.

Dataset	Unsupervised		Supervised			
	Ours	DTW	ST	BOSS	Ensemble	
	Combined				HIVE-COTE	EE
Adiac	0.716	0.604	0.783	0.765	**0.811**	0.665
ArrowHead	0.829	0.703	0.737	0.834	**0.863**	0.811
Beef	0.7	0.633	0.9	0.8	**0.933**	0.633
BeetleFly	0.9	0.7	0.9	0.9	**0.95**	0.75
BirdChicken	0.8	0.75	0.8	**0.95**	0.85	0.8
Car	0.817	0.733	**0.917**	0.833	0.867	0.833
CBF	0.994	0.997	0.974	0.998	**0.999**	0.998
ChlorineConcentration	**0.782**	0.648	0.7	0.661	0.712	0.656
CinCECGTorso	0.74	0.651	0.954	0.887	**0.996**	0.942
Coffee	**1**	**1**	0.964	**1**	**1**	**1**
Computers	0.628	0.7	0.736	0.756	**0.76**	0.708
CricketX	0.777	0.754	0.772	0.736	**0.823**	0.813
CricketY	0.767	0.744	0.779	0.754	**0.849**	0.805
CricketZ	0.764	0.754	0.787	0.746	**0.831**	0.782
DiatomSizeReduction	**0.993**	0.967	0.925	0.931	0.941	0.944
DistalPhalanxOutlineCorrect	0.768	0.717	**0.775**	0.728	0.772	0.728
DistalPhalanxOutlineAgeGroup	0.734	**0.77**	**0.77**	0.748	0.763	0.691
DistalPhalanxTW	0.676	0.59	0.662	0.676	**0.683**	0.647
Earthquakes	**0.748**	0.719	0.741	**0.748**	**0.748**	0.741
ECG200	**0.9**	0.77	0.83	0.87	0.85	0.88
ECG5000	0.936	0.924	0.944	0.941	**0.946**	0.939
ECGFiveDays	**1**	0.768	0.984	**1**	**1**	0.82
ElectricDevices	0.732	0.602	0.747	**0.799**	0.77	0.663
FaceAll	0.802	0.808	0.779	0.782	0.803	**0.849**
FaceFour	0.875	0.83	0.852	**1**	0.955	0.909
FacesUCR	0.918	0.905	0.906	0.957	**0.963**	0.945
FiftyWords	0.78	0.69	0.705	0.705	0.809	**0.82**
Fish	0.88	0.823	**0.989**	**0.989**	**0.989**	0.966
FordA	0.935	0.555	0.971	0.93	**0.964**	0.738
FordB	0.81	0.62	0.807	0.711	**0.823**	0.662
GunPoint	0.993	0.907	**1**	**1**	**1**	0.993
Ham	**0.695**	0.467	0.686	0.667	0.667	0.571
HandOutlines	0.922	0.881	**0.932**	0.903	**0.932**	0.889
Haptics	0.455	0.377	**0.523**	0.461	0.519	0.393
Herring	0.578	0.531	0.672	0.547	**0.688**	0.578
InlineSkate	0.447	0.384	0.373	**0.516**	0.5	0.46
InsectWingbeatSound	0.623	0.355	0.627	0.523	**0.655**	0.595
ItalyPowerDemand	0.925	0.95	0.948	0.909	**0.963**	0.962
LargeKitchenAppliances	0.848	0.795	0.859	0.765	**0.864**	0.811
Lightning2	**0.918**	0.869	0.738	0.836	0.82	0.885
Lightning7	**0.795**	0.726	0.726	0.685	0.74	0.767
Mallat	**0.964**	0.934	**0.964**	0.938	0.962	0.94
Meat	**0.95**	0.933	0.85	0.9	0.933	0.933
MedicalImages	**0.784**	0.737	0.67	0.718	0.778	0.742
MiddlePhalanxOutlineCorrect	0.814	0.698	0.794	0.78	**0.832**	0.784
MiddlePhalanxOutlineAgeGroup	**0.656**	0.5	0.643	0.545	0.597	0.558
MiddlePhalanxTW	**0.61**	0.506	0.519	0.545	0.571	0.513
MoteStrain	0.871	0.835	0.897	0.879	**0.933**	0.883
NonInvasiveFetalECGThorax1	0.91	0.79	**0.95**	0.838	0.93	0.846
NonInvasiveFetalECGThorax2	0.927	0.865	**0.951**	0.901	0.945	0.913
OliveOil	**0.9**	0.833	**0.9**	0.867	**0.9**	0.867
OSULeaf	0.831	0.591	0.967	0.955	**0.979**	0.806
PhalangesOutlinesCorrect	0.801	0.728	0.763	0.772	**0.807**	0.773
Phoneme	0.289	0.228	0.321	0.265	**0.382**	0.305
Plane	0.99	**1**	**1**	**1**	**1**	**1**
ProximalPhalanxOutlineCorrect	0.859	0.784	**0.883**	0.849	0.88	0.808
ProximalPhalanxOutlineAgeGroup	0.854	0.805	0.844	0.834	**0.859**	0.805
ProximalPhalanxTW	**0.824**	0.761	0.805	0.8	0.815	0.766
RefrigerationDevices	0.517	0.464	**0.581**	0.499	0.557	0.437
ScreenType	0.413	0.397	0.52	0.464	**0.589**	0.445
ShapeletSim	0.817	0.65	0.956	**1**	**1**	0.817
ShapesAll	0.875	0.768	0.842	**0.908**	0.905	0.867
SmallKitchenAppliances	0.715	0.643	0.792	0.725	**0.853**	0.696
SonyAIBORobotSurface1	**0.897**	0.725	0.844	0.632	0.765	0.704
SonyAIBORobotSurface2	**0.934**	0.831	**0.934**	0.859	0.928	0.878
StarLightCurves	0.965	0.907	0.979	0.978	**0.982**	0.926
Strawberry	0.946	0.941	0.962	**0.976**	0.97	0.946
SwedishLeaf	0.931	0.792	0.928	0.922	**0.954**	0.915

Table S2: Accuracy scores of the combined version of our method compared with those of DTW (unsupervised), ST and BOSS (supervised) and HIVE-COTE and EE (supervised ensemble methods), on the first 85 UCR datasets (results on the full archive were not available for comparisons). Bold scores indicate the best performing method.

Dataset	Unsupervised		Supervised			
	Ours	DTW	ST	BOSS	Ensemble	
	Combined				HIVE-COTE	EE
Symbols	0.965	0.95	0.882	0.967	**0.974**	0.96
SyntheticControl	0.983	0.993	0.983	0.967	**0.997**	0.99
ToeSegmentation1	0.952	0.772	0.965	0.939	**0.982**	0.829
ToeSegmentation2	0.885	0.838	0.908	**0.962**	0.954	0.892
Trace	**1**	**1**	**1**	**1**	**1**	0.99
TwoLeadECG	**0.997**	0.905	**0.997**	0.981	0.996	0.971
TwoPatterns	**1**	**1**	0.955	0.993	**1**	**1**
UWaveGestureLibraryX	0.811	0.728	0.803	0.762	**0.84**	0.805
UWaveGestureLibraryY	0.735	0.634	0.73	0.685	**0.765**	0.726
UWaveGestureLibraryZ	0.759	0.658	0.748	0.695	**0.783**	0.724
UWaveGestureLibraryAll	0.941	0.892	0.942	0.939	**0.968**	**0.968**
Wafer	0.993	0.98	**1**	0.995	0.999	0.997
Wine	**0.87**	0.574	0.796	0.741	0.778	0.574
WordSynonyms	0.704	0.649	0.571	0.638	0.738	**0.779**
Worms	0.714	0.584	**0.74**	0.558	0.558	0.662
WormsTwoClass	0.818	0.623	**0.831**	**0.831**	0.779	0.688
Yoga	0.878	0.837	0.818	**0.918**	**0.918**	0.879

Table S3: Accuracy scores of the combined version of our method compared with those of ResNet (supervised), TimeNet and RWS (unsupervised), when available. Bold scores indicate the best performing method. "X"s indicate that a score was reported in the original paper, but was either obtained using transferability or on a reversed train / test split of the dataset, thus not comparable to other results for this dataset.

Dataset	Unsupervised	Supervised	Unsupervised	
	Ours	ResNet	TimeNet	RWS
	Combined			
Adiac	0.716	**0.831**	0.565	-
ArrowHead	0.829	**0.84**	-	-
Beef	0.7	**0.767**	-	0.733
BeetleFly	**0.9**	0.85	-	-
BirdChicken	0.8	**0.95**	-	-
Car	0.817	**0.917**	-	-
CBF	**0.994**	0.989	-	-
ChlorineConcentration	0.782	**0.835**	0.723	0.572
CinCECGTorso	0.74	**0.838**	-	-
Coffee	**1**	**1**	-	-
Computers	0.628	**0.816**	-	-
CricketX	0.777	**0.79**	0.659	-
CricketY	0.767	**0.805**	X	-
CricketZ	0.764	**0.831**	X	-
DiatomSizeReduction	**0.993**	0.301	-	-
DistalPhalanxOutlineCorrect	**0.768**	X	X	-
DistalPhalanxOutlineAgeGroup	**0.734**	X	X	-
DistalPhalanxTW	**0.676**	X	X	X
Earthquakes	**0.748**	X	-	-
ECG200	**0.9**	0.87	-	-
ECG5000	**0.936**	0.935	0.934	0.933
ECGFiveDays	**1**	0.99	X	-
ElectricDevices	0.732	**0.735**	0.665	-
FaceAll	0.802	**0.855**	-	-
FaceFour	0.875	**0.955**	-	-
FacesUCR	0.918	**0.955**	-	-
FiftyWords	**0.78**	0.732	-	-
Fish	0.88	**0.977**	-	-
FordA	**0.935**	X	X	-
FordB	**0.81**	X	X	X
GunPoint	**0.993**	**0.993**	-	-
Ham	0.695	**0.8**	-	-
HandOutlines	**0.922**	X	-	X
Haptics	0.455	**0.516**	-	-
Herring	0.578	**0.641**	-	-
InlineSkate	**0.447**	0.378	-	-
InsectWingbeatSound	**0.623**	0.506	-	0.619
ItalyPowerDemand	0.925	0.959	-	**0.969**
LargeKitchenAppliances	0.848	**0.904**	-	0.792
Lightning2	**0.918**	0.77	-	-
Lightning7	0.795	**0.863**	-	-
Mallat	0.964	**0.966**	-	0.937
Meat	0.95	**0.983**	-	-
MedicalImages	**0.784**	0.762	0.753	-
MiddlePhalanxOutlineCorrect	**0.814**	X	X	X
MiddlePhalanxOutlineAgeGroup	**0.656**	X	X	-
MiddlePhalanxTW	**0.61**	X	X	-
MoteStrain	0.871	**0.924**	-	-
NonInvasiveFetalECGThorax1	0.91	**0.946**	-	0.907
NonInvasiveFetalECGThorax2	0.927	**0.944**	-	-
OliveOil	**0.9**	0.867	-	-
OSULeaf	0.831	**0.979**	-	-
PhalangesOutlinesCorrect	0.801	**0.857**	0.772	-
Phoneme	0.289	**0.333**	-	-
Plane	0.99	**1**	-	-
ProximalPhalanxOutlineCorrect	0.859	**0.914**	X	0.711
ProximalPhalanxOutlineAgeGroup	**0.854**	0.839	X	X
ProximalPhalanxTW	**0.824**	X	X	-
RefrigerationDevices	**0.517**	**0.517**	-	-
ScreenType	0.413	**0.632**	-	-
ShapeletSim	0.817	**1**	-	-
ShapesAll	0.875	**0.917**	-	-
SmallKitchenAppliances	0.715	**0.789**	-	-
SonyAIBORobotSurface1	0.897	**0.968**	-	-
SonyAIBORobotSurface2	0.934	**0.986**	-	-
StarLightCurves	0.965	**0.972**	-	-
Strawberry	**0.946**	X	X	-

Table S3: Accuracy scores of the combined version of our method compared with those of ResNet (supervised), TimeNet and RWS (unsupervised), when available. Bold scores indicate the best performing method. "X"s indicate that a score was reported in the original paper, but was either obtained using transferability or on a reversed train / test split of the dataset, thus not comparable to other results for this dataset.

Dataset	Unsupervised	Supervised	Unsupervised	
	Ours	ResNet	TimeNet	RWS
	Combined			
SwedishLeaf	0.931	**0.955**	0.901	-
Symbols	**0.965**	0.927	-	-
SyntheticControl	0.983	**1**	0.983	-
ToeSegmentation1	0.952	**0.969**	-	-
ToeSegmentation2	0.885	**0.915**	-	-
Trace	**1**	**1**	-	-
TwoLeadECG	0.997	**1**	-	-
TwoPatterns	**1**	**1**	0.999	0.999
UWaveGestureLibraryX	**0.811**	0.78	-	-
UWaveGestureLibraryY	**0.735**	0.675	-	-
UWaveGestureLibraryZ	**0.759**	0.75	-	-
UWaveGestureLibraryAll	**0.941**	0.862	-	-
Wafer	0.993	**0.998**	0.994	0.993
Wine	**0.87**	0.611	-	-
WordSynonyms	**0.704**	0.625	-	-
Worms	**0.714**	X	-	-
WormsTwoClass	**0.818**	X	-	-
Yoga	**0.878**	0.857	0.866	-

Table S4: Accuracy scores of variants of our method and of DTW on the remaining 43 UCR datasets, except DodgerLoopDay, DodgerLoopGame and DodgerLoopWeekend which contain missing values. Bold scores indicate the best performing method.

Dataset	Unsupervised							DTW
	Ours							
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	Combined	Combined (1-NN)	FordA	
ACSF1	**0.91**	0.87	0.86	0.9	0.81	0.85	0.73	0.64
AllGestureWiimoteX	0.721	0.746	0.747	0.763	**0.779**	0.736	0.693	0.716
AllGestureWiimoteY	0.741	0.744	0.759	0.726	**0.793**	0.756	0.713	0.729
AllGestureWiimoteZ	0.687	0.697	0.691	0.723	**0.763**	0.716	0.71	0.643
BME	**0.993**	**0.993**	0.987	**0.993**	**0.993**	0.947	0.96	0.9
Chinatown	0.951	0.951	0.942	0.951	**0.962**	0.936	**0.962**	0.957
Crop	0.728	0.726	0.728	0.722	**0.746**	0.695	0.727	0.665
EOGHorizontalSignal	0.552	0.566	0.536	**0.605**	0.588	0.522	0.47	0.503
EOGVerticalSignal	0.398	0.414	0.431	0.434	**0.489**	0.431	0.439	0.448
EthanolLevel	0.418	0.34	0.316	0.382	0.392	0.274	**0.558**	0.276
FreezerRegularTrain	0.986	0.988	0.979	0.956	0.955	0.963	**0.992**	0.899
FreezerSmallTrain	0.967	0.956	0.906	**0.933**	0.928	0.872	0.862	0.753
Fungi	**1**	**1**	**1**	**1**	**1**	**1**	0.925	0.839
GestureMidAirD1	**0.638**	0.577	0.592	0.608	0.615	0.546	0.608	0.569
GestureMidAirD2	0.508	0.515	0.523	**0.546**	0.508	0.415	0.538	0.608
GestureMidAirD3	0.269	**0.331**	0.308	0.285	**0.331**	0.246	0.292	0.323
GesturePebbleZ1	0.826	0.843	0.913	0.919	**0.936**	0.814	0.547	0.791
GesturePebbleZ2	0.861	0.873	0.88	**0.899**	0.88	0.791	0.538	0.671
GunPointAgeSpan	0.984	0.984	**0.994**	**0.994**	0.987	0.991	0.987	0.918
GunPointMaleVersusFemale	**1**	**1**	**1**	0.997	**1**	0.994	**1**	0.997
GunPointOldVersusYoung	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0.838
HouseTwenty	**0.95**	0.933	0.916	0.933	**0.95**	0.924	0.882	0.924
InsectEPGRegularTrain	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0.872
InsectEPGSmallTrain	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0.735
MelbournePedestrian	0.949	0.946	0.943	0.944	**0.951**	0.914	0.947	0.791
MixedShapesRegularTrain	0.916	0.906	0.904	0.905	**0.927**	0.898	0.898	0.842
MixedShapesSmallTrain	0.864	0.857	0.871	0.86	**0.877**	0.829	0.861	0.78
PickupGestureWiimoteZ	0.72	**0.8**	0.78	0.74	0.78	0.72	0.74	0.66
PigAirwayPressure	0.385	0.452	**0.51**	**0.51**	0.486	0.332	0.317	0.106
PigArtPressure	0.88	0.933	**0.942**	0.928	0.933	0.861	0.591	0.245
PigCVP	0.404	0.548	0.62	**0.788**	0.712	0.385	0.534	0.154
PLAID	0.533	0.549	0.574	0.555	0.559	0.696	0.493	**0.84**
PowerCons	**0.961**	0.939	0.9	0.9	0.928	0.894	0.933	0.878
Rock	0.62	0.62	0.58	0.58	**0.68**	0.5	0.54	0.6
SemgHandGenderCh2	0.845	0.852	0.873	0.89	**0.902**	0.863	0.84	0.802
SemgHandMovementCh2	0.711	0.649	0.7	**0.789**	0.784	0.709	0.516	0.584
SemgHandSubjectCh2	0.767	0.816	0.851	0.853	**0.876**	0.72	0.591	0.727
ShakeGestureWiimoteZ	0.92	0.92	**0.94**	0.92	**0.94**	0.86	0.9	0.86
SmoothSubspace	0.933	**0.96**	0.94	**0.96**	0.953	0.833	0.94	0.827
UMD	0.979	0.986	**0.993**	**0.993**	**0.993**	0.958	0.986	**0.993**

S4 Multivariate Time Series

Full results corresponding to the UEA archive datasets for our method as well as the ones of DTW_D as reported by Bagnall et al. (2018) are presented in Table S5, for the unique train / test split provided in the archive.

Table S5: Accuracy scores of variants of our method on all UEA datasets, compared to DTW_D. Bold scores indicate the best performing method.

Dataset	Unsupervised				DTW_D
	Ours				
	$K = 5$	$K = 10$	$K = 20$	Combined	
ArticularyWordRecognition	0.967	0.973	0.943	**0.987**	**0.987**
AtrialFibrillation	**0.2**	0.067	0.133	0.133	**0.2**
BasicMotions	**1**	**1**	**1**	**1**	0.975
CharacterTrajectories	0.986	0.99	0.993	**0.994**	0.989
Cricket	0.958	0.972	0.972	0.986	**1**
DuckDuckGeese	0.6	**0.675**	0.65	**0.675**	0.6
EigenWorms	0.87	0.802	0.84	**0.878**	0.618
Epilepsy	**0.971**	**0.971**	**0.971**	0.957	0.964
Ering	**0.133**	**0.133**	**0.133**	**0.133**	**0.133**
EthanolConcentration	0.289	0.251	0.205	0.236	**0.323**
FaceDetection	0.522	0.525	0.513	0.528	**0.529**
FingerMovements	0.55	0.49	**0.58**	0.54	0.53
HandMovementDirection	0.311	0.297	**0.351**	0.27	0.231
Handwriting	0.447	0.464	0.451	**0.533**	0.286
Heartbeat	**0.756**	0.732	0.741	0.737	0.717
InsectWingbeat	0.159	0.158	0.156	**0.16**	-
JapaneseVowels	0.984	0.986	**0.989**	**0.989**	0.949
Libras	0.878	**0.883**	**0.883**	0.867	0.87
LSST	0.535	0.552	0.509	**0.558**	0.551
MotorImagery	0.53	0.54	**0.58**	0.54	0.5
NATOPS	0.933	0.917	0.917	**0.944**	0.883
PEMS-SF	0.636	0.671	0.676	**0.688**	0.711
PenDigits	**0.985**	0.979	0.981	0.983	0.977
Phoneme	0.216	0.214	0.222	**0.246**	0.151
RacketSports	0.776	0.836	0.855	**0.862**	0.803
SelfRegulationSCP1	0.795	0.826	0.843	**0.846**	0.775
SelfRegulationSCP2	0.55	0.539	0.539	**0.556**	0.539
SpokenArabicDigits	0.908	0.894	0.905	0.956	**0.963**
StandWalkJump	0.333	**0.4**	0.333	**0.4**	0.2
UWaveGestureLibrary	0.884	0.869	0.875	0.884	**0.903**

S5 Discussion of the Choice of Encoder

One of the aims of this work is to propose a representation learning method for time series that is scalable. For this reason, and as explained in Section 4, we did not consider using an LSTM as encoder $\boldsymbol{f}$. Nonetheless, we experimented with such an encoder on a small set of UCR datasets in order to get an indication of its performance versus the proposed encoder in this paper. We use the same optimization hyperparameters as those used to train the causal CNN encoder and choose a two-layer LSTM with hidden size 256 in order to compare both networks with similar computational time and memory usage. Corresponding results are compiled in Table S6.

We observe on this restricted set of experiments that not only the proposed encoder outperforms the LSTM encoder, but it does so by a large margin. This indicates that the proposed causal CNN encoder is more adapted to the considered task and training method.

Table S6: Results of our training method combined with the proposed causal CNN encoder on one hand, and with an LSTM encoder on the other hand ($K = 5$). Bold scores indicate the best performing method.

Dataset	Causal CNN	LSTM
Adiac	**0.703**	0.269
Computers	**0.676**	0.492
CricketX	**0.7**	0.136
DistalPhalanxTW	**0.662**	**0.662**
Earthquakes	**0.748**	**0.748**
HandOutlines	**0.908**	0.646
NonInvasiveFetalECGThorax1	**0.904**	0.169
PhalangesOutlinesCorrect	**0.795**	0.613
RefrigerationDevices	**0.547**	0.411
UWaveGestureLibraryX	**0.806**	0.357
Wafer	**0.993**	0.896