UNDER REVIEW

RESCIENCE C

Replication / NeurIPS 2019 Reproducibility Challenge

[Re] Unsupervised Scalable Representation Learning for Multivariate Time Series

Felix Liljefors[1], Moein Sorkhei[1], Sofia Broomé[1],
[1]KTH Royal Institute of Technology, Stockholm, Sweden

Edited by
Koustuv Sinha

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
15 February 2020

Published
–

DOI
–

1 Introduction

The paper Unsupervised Scalable Representation Learning for Multivariate Time Series by [1] presents an unsupervised approach to learning representations for time series that can be used for subsequent classification. An encoder architecture with dilated causal convolutional blocks is trained to minimize a triplet loss. The triplet loss is based on the idea that a subseries, x_{pos}, of a time series, x_{ref}, should be closer to x_{ref} in representation space than the representation of a randomly sampled time series from the dataset, x_{neg}. This type of loss was first introduced by [2].

To evaluate the strength of the learned representations, an SVM classifier is trained using the labels corresponding to these representations. The evaluation is performed on 159 datasets, together constituting the main benchmarking datasets for time series. The first group of datasets (the UCR Archive, [3]) contains univariate, short (between 15 and 2844 time steps) series; the second group, the UEA dataset ([4]), contains multivariate time series of up to 1345 dimensions and varying sample lengths. The last dataset, the Individual Household Electric Power Consumption (IHEPC) dataset from the UCI Machine Learning Repository ([5]), is a single seven-dimensional time series with more than two million time steps. The authors' intention with using this variety of datasets is to show the universality of their method, which is one of their main claims.

We have re-implemented the authors' method from scratch as best as we could by using only the paper as instruction. It should be noted that a code repository for the article was made public by the authors, but this was not employed for the purpose of investigating the replicability of the work starting from scratch. This means that we adhere to the Replication track of the Reproducibility challenge for NeurIPS 2019. Our implementation, as well as the authors', was made using the Pytorch ([6]) library and can be found at https://github.com/lilfelix/reproducibility_NeurIPS19.

The main motivation to do a full replication study, avoiding the use of readily provided code, is to better be able to detect if there are parts of the implementation that are crucial for the results, but not presented as such in the original paper. This risk exists generally in machine learning ([7]), and has in particular been brought to light for deep learning (e.g. [8], [9]), due to the latter's typically vast number of hyperparameters to tune.

1.1 Target questions

In this report, we mainly address the following questions.

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

- Can the classification and regression results be reproduced using only the article's description of the method?
- Can we reproduce the claimed transferability of the method and its proposed success on sparsely labeled datasets?
- Which missing instructions for the experiments might, if relevant, have affected the results?
- Which 'hidden' assumptions, that mattered for the results, were made by the authors?

1.2 Additional contributions

We present results for the authors' method on three additional datasets from the UCR Archive: DodgerLoopDay, DodgerLoopGame and DodgerLoopWekend. These results can be found in Table 6 of the supplementary material.
The rest of the article is organized as follows: Section 2 presents the implementation details and the results, in Section 3 we discuss the findings, and Section 4 is dedicated to the conclusions.

2 Experimental methodology and implementation details

In Sections 2.1-2.5, we will group the reproducibility experiments into the three dataset groups (UCR, UEA and IHEPC). First, we will describe our general experimental methodology for this study and some assumptions not mentioned in the original article that were necessary to make for all three groups of datasets. A strength of the original paper is that experimental hyperparameters are listed to a greater detail than is typically done in deep learning papers. Nevertheless, there are a number of design choices of importance for the implementation that were not mentioned in the article.
We implemented Algorithm 1 from the paper (described in pseudo-code), to sample three subseries[1] x_{ref}, x_{pos} and x_{neg}, used to compute the triplet loss. This algorithm is described by the authors as generally applicable to all datasets. However, they also mention a variation of Algorithm 1, which they claim speeds up the training time for datasets with fixed length time series, yet produces no noticeable difference in the computed loss. We did not implement this variation for fixed length time series, but instead used Algorithm 1 for all experiments. In the case of multivariate datasets, we assume that the length of a subseries, e.g. size(x_{ref}), is the same across all dimensions of the time series being sampled from. In other words, when sampling x_{ref} from some d-dimensional time series y, size(x^i_{ref}) is the same for all $i \in [1, d]$.
Notably, except for IHEPC, the authors state that they have used a batch size of 10 throughout their experiments. Mini-batch training requires sequence padding when there are samples of different lengths. This is the case for these experiments since subseries are explicitly sampled at different lengths from the dataset (see Algorithm 1 in [1]). However, there is no mention of how this padding is carried out in the article. For our experiments, we zero-pad the samples to the maximal sample length for each batch.
Another training detail not made explicit in the article is whether the representations used for classification should be obtained from full samples of the datasets (i.e. of length T, if T is the number of time steps for one sample), or rather from randomly sampled subseries of the samples (i.e. of length T', where $T' \in [1, T]$). For our experiments, we used the full samples (length T) in order to use the training set maximally when training the classifier.

[1] A subseries here is a contiguous sub-sequence of a time series.

Regarding the SVM training, a hyperparameter optimization is performed for the weighting C of the error term using cross-validation across the training set. The authors indicate that they avoided this cross-validation for datasets with a small number of training samples or datasets with few training samples per class. In lack of an exact list of which datasets they refer to, we avoided the hyperparameter search only for the datasets that fell below the size limit set by the cross-validation tool of scikit-learn ([10]).

2.1 The UCR Time Series Classification

The UCR Archive contains 128 univariate datasets. Each dataset consists of training and test time series which may have different lengths and missing values. The paper's authors decided to not use the three datasets with missing values (DodgerLoopDay, DodgerLoopGame and DodgerLoopWekend). However, the UCR Archive contains modified versions of these datasets with linearly interpolated values that we used. Our motivation for this is that the UCR Archive briefing document recommends that all available datasets are used, to allow for comprehensive comparisons and avoid cherry-picking results.
For the task of time series classification on the UCR Archive datasets, the authors consider other state-of-the art methods, some of which are based on neural networks. More specifically, they compare their method to neural network methods, both unsupervised (TimeNet, [11]; RWS, [12]) and supervised (ResNet, [13]), and non-neural network methods, both unsupervised (Dynamic Time Warping (DTW)) and supervised (HIVE-COTE, [14]; ST, [15]; BOSS, [16]; EE, [17]). The use of ResNet as a baseline in the paper is motivated by referencing an extensive review made by [18], who found it to be the best supervised neural network method for time series classification.
In Table 1 of the paper, the authors present their accuracy scores on 6 datasets from the UCR Archive, together with baseline[2] accuracy scores from non-neural network models (both supervised and unsupervised). In a similar fashion, Table 1 in this report presents our replicated accuracy scores for the same datasets, together with the original results from the paper for comparison. We also introduce a comparison of our replicated scores and the authors' against neural network models in Table 2.
Table 1 and 2 present the highest accuracy for each dataset, across all values of $K \in \{1, 2, 5, 10\}$. Results for all values of K and all datasets can be found in Table 6 of the supplementary material. Further, these tables contain accuracy scores for three additional UCR datasets (not presented in the original paper's Table 1), for which we also have baseline scores of both RWS and TimeNet (see Table 2). These are presented below the dashed line, and provide a comparison between the non-neural network scores in Table 1 and the neural network scores in Table 2.

Table 1. Our best achieved scores compared to the authors' best achieved scores along with the scores for all the non-neural-network methods for some UCR datasets.

			Baseline				
Dataset	Ours	Authors'	DTW	ST	BOSS	HIVE-COTE	EE
ECGFiveDays	1	1	1	0.984	1	1	0.82
DiatomSizeReduction	0.987	0.993	0.967	0.925	0.931	0.941	0.944
FordB	0.785	0.81	0.62	0.807	0.711	0.823	0.662
Ham	0.762	0.724	0.467	0.686	0.667	0.667	0.571
Phoneme	0.225	0.289	0.228	0.321	0.265	0.382	0.305
SwedishLeaf	0.922	0.931	0.792	0.928	0.922	0.954	0.915
ECG5000	0.928	0.94	0.924	0.944	0.941	0.946	0.939
TwoPatterns	1	1	1	0.955	0.993	1	1
Wafer	0.998	0.995	0.98	1	0.995	0.999	0.997

[2]The authors adapted the results from http://www.timeseriesclassification.com/singleTrainTest.csv, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/, and https://github.com/hfawaz/dl-4-tsc/blob/master/results/results-uea.csv

Table 2. Our best achieved scores compared to the authors' best achieved scores along with the scores for supervised and unsupervised neural network methods for some UCR datasets. Missing scores are indicated by '-'.

			Baseline		
Dataset	Ours	Authors'	ResNet	TimeNet	RWS
ECGFiveDays	1	1	0.99	-	-
DiatomSizeReduction	0.987	0.993	0.301	-	-
FordB	0.785	0.81	-	-	-
Ham	0.762	0.724	0.8	-	-
Phoneme	0.225	0.289	0.333	-	-
SwedishLeaf	0.922	0.931	0.955	0.901	-
ECG5000	0.928	0.94	0.935	0.934	0.933
TwoPatterns	1	1	1	0.999	0.999
Wafer	0.998	0.995	0.998	0.994	0.993

2.2 Sparse labeling experiment

Figure 1 shows our reproduction of the sparse labeling experiment of Section 5.1.1 of the paper, where a comparison is made between the authors' method and ResNet when training on fractions of the labeled data from a randomly chosen dataset (TwoPatterns). Just as in the original article, there is a striking difference between the ResNet and the encoder model from the article on the smaller portions ($< 50\%$) of training data. Hence our results are in line with the authors' claim; an SVM trained on labeled representations from their architecture performs better than ResNet trained on the raw data when training data is sparsely labeled.

The shapes of the ResNet graphs (in blue and red in Figure 1) are different. This is likely explained by the fact that, surprisingly, there are no details about which ResNet architecture was used for this experiment, or with which set of hyperparameters, or whether it had been pre-trained in any capacity. It is possible that this is made clear in the public code repository. Yet, for the sake of the replications track, we did not look this up.

Instead, for this experiment, we chose to train the least deep among the standard ResNets provided by PyTorch: ResNet-18. This choice was made considering the low dimensionality of the time series data compared to image data which ResNet is typically trained on. Specifically, this experiment was only run on univariate data from the UCR dataset, which further motivated our choice of an as-simple-as-possible ResNet architecture. We trained the network from scratch since it required modified input dimensions compared to the standard ResNet from the Pytorch library.

2.3 Transferability experiment

In the supplementary material, we reproduced the transferability experiment from the original article. These results are shown in Table 6 of the supplementary material, in the column entitled FordA. The idea for the transferability was to train an encoder on the FordA dataset, then pass samples from the other datasets through this encoder and see how well the SVM can learn from training on those representations with labels. The idea with the experiment was to show the generality of such a learned encoder and to show that classification results are comparable even if the encoder is trained on one dataset and produces representations on other datasets. Our results are very similar to the authors' results in Table S1 of the article.

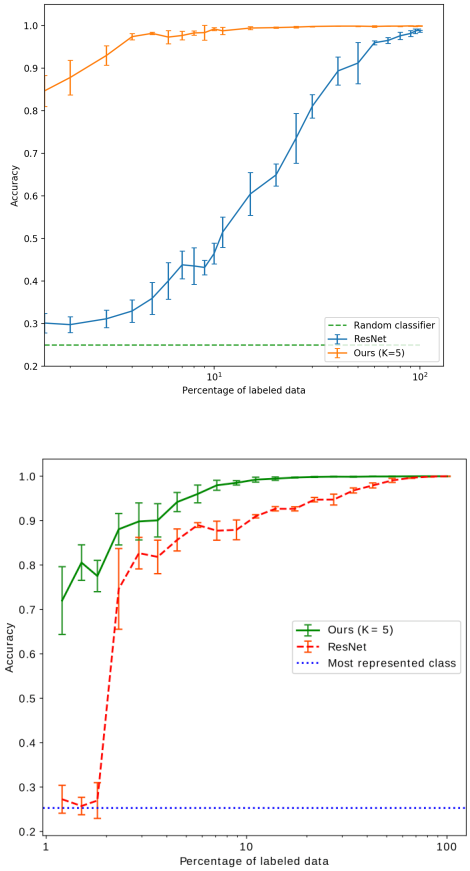

Figure 1. Reproduction of Figure 4 from the original article (right). The same striking difference between ResNet and the authors' method is observed when training on a limited amount of labeled data.

2.4 The UEA Time Series Classification

The UEA archive contains 30 multivariate datasets, with the dimensions per sample ranging from 2 to 1345. Some of these datasets have missing values. While the handling of missing values was not described in the paper, we chose to linearly interpolate these values before training the encoder. For the four datasets with the longest samples (EigenWorms, EthanolConcentration, MotorImagery and StandWalkJump) we needed to adjust the batch size to below 10 in order to fit the GPU memory. These adjustments can be seen in Table 3.

Table 3. Adjusted batch sizes for datasets with the longest sample lengths

Dataset	EigenWorms	EthanolConcentration	MotorImagery	StandWalkJump
$K = 5$	4	10	6	8
$K = 10$	2	10	6	8
$K = 20$	1	8	6	8

The authors compare their scores on the UEA datasets to that of $\mathrm{DTW_D}$. DTW_D, an extension of DTW, is the best baseline known in the multivariate setting, according to the paper. In Table 4, we present our best achieved accuracy (taken among results using different values of K) next to the authors' scores, along with the scores of $\mathrm{DTW_D}$ for a

random selection of UEA datasets.
As can be seen, our best scores closely match the best scores of the authors for these datasets. Our full scores on all the UEA datasets except InsectWingBeat can be found in Table 7 of the supplementary material. InsectWingBeat with its large number of training samples was unfeasible for our time horizon to train the SVM with when using scikit-learn, see the discussion in Section 3. For $K = 10$, the training would have taken more than 12 hours, so we decided to not include this dataset.

Table 4. Best scores achieved in our experiments compared to the authors' best scores along with the $\mathrm{DTW_D}$ method on some UEA datasets

Dataset	Ours	Authors'	Baseline $\mathrm{DTW_D}$
ArticularyWordRecognition	0.983	0.987	0.987
AtrialFibrillation	0.333	0.2	0.2
Libras	0.889	0.883	0.87
NATOPS	0.922	0.944	0.883
UWaveGestureLibrary	0.903	0.884	0.903

2.5 The IHEPC experiment

The Individual Household Electric Power Consumption (IHEPC) is a seven-dimensional time series (not including the temporal dimension) containing over two million measurements. These measurements were split by the authors into train and test sets of 500,000 and ∼1,500,000 values, respectively. The paper does not mention the multivariate property of the IHEPC dataset and describes that the data was normalized in the same way as the univariate UCR datasets (multivariate normalization is described separately). Further, the plotted result for the IHEPC experiment (Figure 5 in the paper) shows only one feature (active power consumption), leading us to believe that only this feature was used. This belief was also strengthened when we trained the encoder on a single time series of length 500,000 measurements, since using all seven features simply requires more GPU memory than the authors specified in the report (16 GB). Lastly, the dataset has missing values which we chose to linearly interpolate, as the paper does not specify how these were handled.
The paper mentions that the encoder took no more than a few hours to train on a single Nvidia Tesla P100 GPU. For us it took 10.5 hours to complete the 400 optimization steps on a Tesla T4 GPU which has the same memory capacity and notably a higher computing performance than the P100. However, it is noteworthy that the triplet loss stopped improving after ∼50 optimization steps (reaching a loss of 4.5 from an initial loss of 800). After 50 optimization steps, the loss fluctuated up and down, but never went lower than 4.5.
The information regarding how the encoder was used for a regression task after being trained on IHEPC was quite sparse in the paper, which led us to spend a lot of time determining the most feasible procedure. What we concluded was the following. The encoder is first trained for 400 optimization steps on a single univariate long time series (500,000 measurements), using the feature Global active power. Next, the trained encoder is used to generate two tuples of representations which we denote $\{\mathrm{R}_{day}^{train}, \mathrm{R}_{day}^{test}\}$ and $\{\mathrm{R}_{quarter}^{train}, \mathrm{R}_{quarter}^{test}\}$.
To generate R_{day}^{train}, the train set is split into subseries of length 1440, corresponding to the number of measurements in a day, extracted as sliding windows with stride one (a day window). Each day window is then passed through the encoder to generate a representation. Next a label for each representation is computed as the difference of the mean measurements in the previous and subsequent day window, producing a set of labels L_{day}^{train} for R_{day}^{train}.

Finally a linear regressor is trained on $\{\mathrm{R}_{day}^{train}, \mathrm{L}_{day}^{train}\}$, and tested on $\{\mathrm{R}_{day}^{test}, \mathrm{L}_{day}^{test}\}$, using minimization of MSE as objective. An analogous experiment can also be performed using subseries of length $12{\cdot}7{\cdot}1440$ corresponding to a quarter window. This experiment thus involves training and testing another regressor on $\{\mathrm{R}_{quarter}^{train}, \mathrm{L}_{quarter}^{train}\}, \{\mathrm{R}_{quarter}^{test}, \mathrm{L}_{quarter}^{test}\}$. The only description in the paper concerning the linear regressors come from the following quote:

> 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 on the previously computed representations.

Therefore, we made the following assumptions. We used the same optimizer (including its hyperparameters, such as learning rate) for the regressors as we did for the encoder, i.e. Adam [19]. Further, we trained the regressors for 2000 optimization steps (the default number of optimization steps specified in the paper). Since the regressor is trained on representations of the encoder, we had to encode all windows (either day or quarter) of both the train and test time series ($\sim$ 2 million measurements in total). To measure the efficiency of the representations, the authors compared the regressors trained on representations with regressors trained on the raw, actual values in the IHEPC dataset. It is noteworthy that the train and test datasets consisting of representations are orders of magnitude smaller in size than the raw-valued datasets. The number of scalars representing an encoded window (either day or quarter) is 80, which is the output dimension of the encoder. In contrast, the length of each window of raw values is 1440 and $7 \cdot 12 \cdot 1440$ for day and quarter windows, respectively. We replicated the experiment by training and testing regressors on both the representations and the raw values, and the results are summarized in Table 5. We observed similar time efficiency when evaluating the regressors as the authors reported. As explained by the authors in their original report, the large discrepancy in the wall time when using representations versus raw values on the quarter task is due to raw-valued windows having much larger size than their corresponding representations. For the raw regressors, we observed test MSEs similar to those reported by the authors. For the regressors trained on the representations however, we saw the training loss drop significantly after only a few optimization steps. These regressors also produced much lower MSEs compared to the those trained on raw values, which was unexpected. The authors didn't report such a discrepancy in test MSE between representations and raw values, and we aren't sure what caused this result in our experiment. In summary, our results are not completely aligned with the numbers reported by the authors, yet we saw that the simple linear regressor was more successful in predicting the labels when trained on windows with compact representations.

Table 5. Replicated results on regression task of IHEPC experiment

Task	Metric	Our Representations	Raw values
Day	Test MSE	$40.64 \cdot 10^{-5}$	$2.02 \cdot 10^{-2}$
	Wall time	8.67s	9min 39s
Quarter	Test MSE	$6.21 \cdot 10^{-5}$	$13.28 \cdot 10^{-2}$
	Wall time	1min 38s	31min 27s

3 Discussion of findings

The main advantage of the authors' method is that it is flexible with respect to sequence lengths and domains. Time series with different lengths and dimensionality can be embedded by the encoder architecture. In the bulk of the many experiments, our results closely match those of the authors, both in the univariate and multivariate datasets. In order to formally measure the difference between the results of our experiments and the authors' experiments, the difference between our scores and those of the authors was computed for each possible value of K, for each dataset in both the UCR and UEA Archives. Then we computed the average difference by taking the average of the differences for each possible value of K (including the FordA score) in each dataset.

We observe that the average difference is less than 5% in 68%, less than 8% in 80% and less than 10% in 88% of the UCR datasets. The corresponding numbers for UEA are: less than 5% in 69%, less than 8% in 76% and less than 10% in 76% of the UEA datasets. An offset is expected since the experiments were only run once for each dataset, as in the article. There is stochasticity in the sampling of the sub-sequences during the training of the encoder. These differences seem to be within the expected range and indicate that we have achieved quite similar results without using their implementation. Please note that our focus is to compare our results with the authors' results only to ensure reproducibility of the experiments, rather than comparing our scores with the state-of-the-art methods. Since we have achieved very similar results to those of the authors, we believe that the distribution of rankings of different methods also generally holds true in our experiments. Thus, for the first target question we conclude that the classification results can be reproduced.

Regarding the main advantages of the authors' model over other state-of-the-art supervised methods, we observed in the FordA experiment that the representations generated by the encoder trained on the FordA dataset can be applied for classification in other datasets and achieve acceptable results. This provides support to the authors' claim that the representations generated by such an encoder are transferable and can be applied to classification tasks for other datasets.

Furthermore, we observed in our experiment with the ResNet model that in the case of datasets with limited labeled samples, the authors' unsupervised method can perform significantly better than the ResNet model. This provides support for the claim of performing well with sparse labeled data.

The two last target questions concern missing instructions and hidden assumptions and whether these have a crucial influence on the results. We were able to reproduce the experiments conducted with the UCR and UEA datasets using the paper's instructions, while the long time series experiment (IHEPC dataset) proved more difficult. The long time series experiment lacked many details in its methodology, which led us to make multiple assumptions in order to produce results. For the UCR and UEA experiments, the authors motivated the use of SVMs in the classification step by claiming it allows for efficient training (a matter of minutes in most cases). This held true for the majority of the datasets in our replication study, but for the datasets with the largest number of samples, the training of SVMs took several hours. In the IHEPC experiment, we observed similar efficiency in terms of the wall times when evaluating the regressors on the test set, and we observed similar test errors when training the regressors on the raw values. For the regressors trained on the representations however, we observed lower test errors compared to the numbers reported by the authors. We also saw the training converging much faster with a significant drop in loss after a few optimization steps.

Finally, the fact that we managed to reproduce most results indicates that any hidden assumptions made by the authors were not crucial in the end, unless we accidentally made the exact same set of assumptions, which does not seem likely.

4 Conclusions

In this work, we have presented a replication study of the work by [1] and found that most of the results are replicable. We have reproduced almost a thousand[3] (942) results from the original article and they largely follow the results obtained by the authors. Since our experiments were run with a set of additional assumptions, this speaks in favor of the robustness of the authors' method.

References

1. J. Franceschi, A. Dieuleveut, and M. Jaggi. "Unsupervised Scalable Representation Learning for Multivariate Time Series." In: **CoRR** abs/1901.10738 (2019). arXiv: 1901.10738.
2. F. Schroff, D. Kalenichenko, and J. Philbin. "Facenet: A unified embedding for face recognition and clustering." In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2015, pp. 815–823.
3. H. A. Dau et al. **The UCR Time Series Classification Archive**. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. Oct. 2018.
4. A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. "The UEA multivariate time series classification archive, 2018." In: **arXiv preprint arXiv:1811.00075** (2018).
5. D. Dua and C. Graff. **UCI Machine Learning Repository**. 2017.
6. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic Differentiation in PyTorch." In: **NIPS Autodiff Workshop**. 2017.
7. Z. C. Lipton and J. Steinhardt. "Troubling Trends in Machine Learning Scholarship." In: **Queue** 17.1 (Feb. 2019), 80:45–80:77.
8. P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. **Deep Reinforcement Learning that Matters**. cite arxiv:1709.06560Comment: Accepted to the Thirthy-Second AAAI Conference On Artificial Intelligence (AAAI), 2018. 2017.
9. G. Melis, C. Dyer, and P. Blunsom. "On the State of the Art of Evaluation in Neural Language Models." In: **CoRR** abs/1707.05589 (2017). arXiv: 1707.05589.
10. F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: **Journal of Machine Learning Research** 12 (2011), pp. 2825–2830.
11. P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff. "TimeNet: Pre-trained deep recurrent neural network for time series classification." In: **CoRR** abs/1706.08838 (2017). arXiv: 1706.08838.
12. L. Wu, I. E.-H. Yen, J. Yi, F. Xu, Q. Lei, and M. Witbrock. "Random Warping Series: A Random Features Method for Time-Series Embedding." In: **International Conference on Artificial Intelligence and Statistics**. 2018, pp. 793–802.
13. K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." In: **arXiv preprint arXiv:1512.03385** (2015).
14. J. Lines, S. Taylor, and A. Bagnall. "HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles for Time Series Classification." In: **2016 IEEE 16th International Conference on Data Mining (ICDM)**. Dec. 2016, pp. 1041–1046.
15. A. Bostrom and A. Bagnall. "Binary Shapelet Transform for Multiclass Time Series Classification." In: **Big Data Analytics and Knowledge Discovery**. Ed. by S. Madria and T. Hara. Cham: Springer International Publishing, 2015, pp. 257–269.
16. P. Schäfer. "The BOSS is Concerned with Time Series Classification in the Presence of Noise." In: **Data Min. Knowl. Discov.** 29.6 (Nov. 2015), pp. 1505–1530.
17. J. Lines and A. Bagnall. "Time series classification with ensembles of elastic distance measures." In: **Data Mining and Knowledge Discovery** 29.3 (May 2015), pp. 565–592.
18. H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. "Deep learning for time series classification: a review." In: **Data Mining and Knowledge Discovery** 33.4 (2019), pp. 917–963.
19. D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization." In: **arXiv preprint arXiv:1412.6980** (2014).

[3] 942 = 128*5 + 24*5*2 + 20*3 + 2 (4 values of K plus FordA transferability for each of the 128 UCR datasets, 5 runs at 24 different fractions of the training data on the TwoPatterns dataset for two models, 3 values of K for the 20 multivariate datasets from UEA, one experiment for days and one for quarters for IHEPC).

Supplementary material for [Re] Unsupervised Representation Learning for Multivariate Timeseries

In this section, we present the full results of our experiments on the UCR and UEA Archives.

5 UCR Archive Results

Table 6. Test accuracy scores for all 128 UCR datasets.

Dataset	K=1	K=2	K=5	K=10	K=combined	FordA (K=5)
ACSF1	0.74	0.82	0.81	0.74	0.74	0.75
Adiac	0.739	0.726	0.716	0.721	0.76	0.762
AllGestureWiimoteX	0.679	0.65	0.657	0.677	0.67	0.666
AllGestureWiimoteY	0.681	0.676	0.686	0.69	0.716	0.706
AllGestureWiimoteZ	0.64	0.674	0.654	0.647	0.699	0.68
ArrowHead	0.777	0.777	0.834	0.771	0.76	0.76
Beef	0.733	0.367	0.667	0.7	0.7	0.733
BeetleFly	0.75	0.75	0.55	0.55	0.75	0.85
BirdChicken	0.75	0.75	0.75	0.85	0.75	0.75
BME	0.953	0.813	0.92	0.953	0.98	0.987
Car	0.75	0.817	0.8	0.783	0.8	0.8
CBF	0.99	0.986	0.94	0.94	0.989	0.98
Chinatown	0.781	0.942	0.843	0.921	0.883	0.965
ChlorineConcentration	0.649	0.606	0.64	0.594	0.7	0.629
CinCECGTorso	0.693	0.775	0.7	0.758	0.799	0.638
Coffee	1	0.929	0.964	1	1	1
Computers	0.656	0.692	0.696	0.7	0.68	0.68
CricketX	0.703	0.674	0.664	0.649	0.736	0.651
CricketY	0.597	0.603	0.597	0.582	0.656	0.6
CricketZ	0.692	0.641	0.685	0.633	0.721	0.682
Crop	0.725	0.721	0.711	0.716	0.74	0.719
DiatomSizeReduction	0.922	0.951	0.987	0.915	0.944	0.98
DistalPhalanxOutlineAgeGroup	0.748	0.748	0.734	0.741	0.748	0.719
DistalPhalanxOutlineCorrect	0.79	0.772	0.797	0.772	0.775	0.775
DistalPhalanxTW	0.691	0.633	0.683	0.676	0.662	0.655
DodgerLoopDay	0.55	0.5	0.475	0.488	0.488	0.475
DodgerLoopGame	0.891	0.804	0.797	0.855	0.862	0.848
DodgerLoopWeekend	0.884	0.957	0.848	0.754	0.884	0.957
Earthquakes	0.748	0.748	0.748	0.748	0.748	0.748
ECG200	0.9	0.85	0.83	0.86	0.91	0.89
ECG5000	0.922	0.91	0.923	0.914	0.928	0.917
ECGFiveDays	0.998	1	1	0.997	1	0.956
ElectricDevices	0.697	0.68	0.679	0.691	0.701	0.665
EOGHorizontalSignal	0.547	0.511	0.494	0.569	0.536	0.481
EOGVerticalSignal	0.401	0.409	0.42	0.403	0.434	0.425
EthanolLevel	0.374	0.4	0.388	0.426	0.412	0.476
FaceAll	0.78	0.827	0.756	0.773	0.8	0.766
FaceFour	0.477	0.682	0.545	0.773	0.75	0.727
FacesUCR	0.836	0.84	0.804	0.818	0.86	0.815
FiftyWords	0.721	0.721	0.723	0.725	0.736	0.708
Fish	0.914	0.857	0.834	0.84	0.88	0.874

Table 6. Test accuracy scores for all 128 UCR datasets.

Dataset	K=1	K=2	K=5	K=10	K=combined	FordA (K=5)
FordA	0.898	0.914	0.898	0.897	0.918	0.899
FordB	0.778	0.744	0.762	0.751	0.785	0.769
FreezerRegularTrain	0.986	0.991	0.989	0.982	0.989	0.993
FreezerSmallTrain	0.811	0.741	0.783	0.887	0.791	0.916
Fungi	0.742	0.871	0.941	0.962	0.941	0.871
GestureMidAirD1	0.538	0.546	0.615	0.631	0.569	0.615
GestureMidAirD2	0.515	0.523	0.446	0.485	0.538	0.5
GestureMidAirD3	0.3	0.254	0.331	0.323	0.292	0.323
GesturePebbleZ1	0.645	0.564	0.529	0.622	0.669	0.535
GesturePebbleZ2	0.563	0.791	0.551	0.57	0.709	0.595
GunPoint	0.933	0.98	0.973	0.987	0.98	0.973
GunPointAgeSpan	0.94	0.959	0.937	0.908	0.949	0.949
GunPointMaleVersusFemale	1	1	1	1	1	0.994
GunPointOldVersusYoung	1	1	1	1	1	1
Ham	0.743	0.657	0.59	0.562	0.762	0.676
HandOutlines	0.932	0.916	0.916	0.93	0.935	0.916
Haptics	0.484	0.474	0.477	0.442	0.477	0.474
Herring	0.547	0.5	0.641	0.594	0.578	0.594
HouseTwenty	0.832	0.874	0.849	0.891	0.882	0.824
InlineSkate	0.367	0.404	0.387	0.431	0.407	0.442
InsectEPGRegularTrain	1	1	1	1	1	1
InsectEPGSmallTrain	1	1	1	1	1	1
InsectWingbeatSound	0.599	0.563	0.57	0.548	0.605	0.567
ItalyPowerDemand	0.934	0.941	0.927	0.954	0.946	0.913
LargeKitchenAppliances	0.752	0.784	0.816	0.76	0.776	0.765
Lightning2	0.852	0.689	0.82	0.754	0.836	0.836
Lightning7	0.753	0.699	0.74	0.699	0.712	0.644
Mallat	0.93	0.906	0.939	0.946	0.974	0.953
Meat	0.917	0.85	0.9	0.917	0.9	0.933
MedicalImages	0.754	0.717	0.709	0.708	0.757	0.724
MelbournePedestrian	0.285	0.283	0.29	0.288	0.286	0.299
MiddlePhalanxOutlineAgeGroup	0.656	0.643	0.656	0.662	0.636	0.63
MiddlePhalanxOutlineCorrect	0.801	0.808	0.828	0.794	0.814	0.818
MiddlePhalanxTW	0.558	0.578	0.571	0.584	0.565	0.597
MixedShapesRegularTrain	0.918	0.911	0.908	0.917	0.919	0.912
MixedShapesSmallTrain	0.862	0.844	0.876	0.826	0.868	0.868
MoteStrain	0.779	0.802	0.823	0.83	0.81	0.813
NonInvasiveFetalECGThorax1	0.921	0.907	0.895	0.902	0.922	0.926
NonInvasiveFetalECGThorax2	0.931	0.926	0.922	0.916	0.933	0.924
OliveOil	0.8	0.833	0.767	0.8	0.8	0.833
OSULeaf	0.719	0.607	0.702	0.678	0.744	0.653
PhalangesOutlinesCorrect	0.773	0.797	0.787	0.793	0.759	0.788
Phoneme	0.207	0.205	0.207	0.192	0.225	0.187
PickupGestureWiimoteZ	0.54	0.66	0.74	0.58	0.7	0.62
PigAirwayPressure	0.216	0.207	0.221	0.236	0.226	0.269
PigArtPressure	0.736	0.615	0.663	0.663	0.702	0.774
PigCVP	0.495	0.481	0.476	0.481	0.519	0.577
PLAID	0.46	0.475	0.471	0.466	0.495	0.471
Plane	0.99	1	0.99	0.981	0.99	0.99
PowerCons	0.922	0.906	0.939	0.933	0.939	0.939
ProximalPhalanxOutlineAgeGroup	0.834	0.844	0.854	0.849	0.834	0.863
ProximalPhalanxOutlineCorrect	0.859	0.873	0.856	0.859	0.869	0.859

Table 6. Test accuracy scores for all 128 UCR datasets.

Dataset	K=1	K=2	K=5	K=10	K=combined	FordA (K=5)
ProximalPhalanxTW	0.785	0.81	0.82	0.82	0.815	0.81
RefrigerationDevices	0.515	0.525	0.504	0.504	0.541	0.536
Rock	0.56	0.62	0.76	0.64	0.66	0.5
ScreenType	0.381	0.443	0.44	0.411	0.467	0.424
SemgHandGenderCh2	0.84	0.848	0.817	0.795	0.828	0.85
SemgHandMovementCh2	0.636	0.607	0.593	0.569	0.667	0.578
SemgHandSubjectCh2	0.758	0.773	0.709	0.716	0.764	0.702
ShakeGestureWiimoteZ	0.78	0.72	0.86	0.8	0.84	0.9
ShapeletSim	0.511	0.522	0.561	0.594	0.589	0.55
ShapesAll	0.812	0.798	0.825	0.823	0.84	0.838
SmallKitchenAppliances	0.691	0.696	0.717	0.704	0.723	0.707
SmoothSubspace	0.947	0.92	0.98	0.973	0.973	0.947
SonyAIBORobotSurface1	0.745	0.837	0.651	0.659	0.739	0.752
SonyAIBORobotSurface2	0.793	0.795	0.799	0.869	0.842	0.811
StarLightCurves	0.968	0.97	0.97	0.964	0.971	0.968
Strawberry	0.946	0.914	0.935	0.932	0.938	0.957
SwedishLeaf	0.906	0.912	0.91	0.907	0.922	0.898
Symbols	0.908	0.886	0.847	0.886	0.923	0.941
SyntheticControl	0.983	0.997	0.977	0.98	0.99	0.99
ToeSegmentation1	0.864	0.781	0.816	0.864	0.886	0.816
ToeSegmentation2	0.8	0.877	0.838	0.785	0.854	0.808
Trace	1	1	1	1	1	1
TwoLeadECG	0.971	0.792	0.965	0.857	0.935	0.985
TwoPatterns	1	1	1	1	1	1
UMD	0.986	0.882	0.972	0.979	1	0.972
UWaveGestureLibraryAll	0.934	0.916	0.918	0.9	0.951	0.908
UWaveGestureLibraryX	0.805	0.795	0.786	0.79	0.808	0.775
UWaveGestureLibraryY	0.716	0.718	0.714	0.708	0.736	0.702
UWaveGestureLibraryZ	0.736	0.731	0.714	0.729	0.752	0.732
Wafer	0.994	0.991	0.994	0.995	0.998	0.995
Wine	0.611	0.5	0.5	0.5	0.5	0.5
WordSynonyms	0.621	0.589	0.6	0.619	0.65	0.599
Worms	0.558	0.545	0.61	0.519	0.623	0.597
WormsTwoClass	0.662	0.636	0.662	0.61	0.662	0.636
Yoga	0.828	0.8	0.814	0.781	0.859	0.786

6 UEA Archive Results

The following is our full results on the 30 UEA datasets. Note that we were not able to reproduce the SVM results for the InsectWingbeat dataset since it has a very large number of samples (30,000 training and 20,000 test samples), and we could not manage to train the SVM on it in a timely manner (training the SVM was taking more than a day using the scikit-learn library.)

Table 7. Test accuracy for all the datasets in the UEA archive.

Dataset	K=5	K=10	K=20	K=combined
ArticularyWordRecognition	0.957	0.963	0.957	0.983
AtrialFibrillation	0.333	0.333	0.333	0.2
BasicMotions	0.925	0.975	0.925	0.975
CharacterTrajectories	0.985	0.985	0.985	0.989
Cricket	0.972	0.903	0.972	0.944

Table 7. Test accuracy for all the datasets in the UEA archive.

Dataset	K=5	K=10	K=20	K=combined
DuckDuckGeese	0.4	0.42	0.42	0.38
EigenWorms	0.641	0.595	0.366	0.55
Epilepsy	0.906	0.942	0.884	0.957
ERing	0.796	0.711	0.548	0.815
EthanolConcentration	0.3	0.266	0.304	0.3
FaceDetection	0.512	0.534	0.531	0.524
FingerMovements	0.54	0.55	0.47	0.5
HandMovementDirection	0.243	0.216	0.311	0.243
Handwriting	0.313	0.347	0.339	0.396
Heartbeat	0.717	0.727	0.741	0.732
InsectWingbeat				
JapaneseVowels	0.951	0.973	0.954	0.962
Libras	0.883	0.856	0.883	0.889
LSST	0.397	0.405	0.386	0.398
MotorImagery	0.58	0.49	0.46	0.51
NATOPS	0.922	0.9	0.911	0.922
PEMS-SF	0.636	0.676	0.642	0.647
PenDigits	0.983	0.981	0.979	0.985
PhonemeSpectra	0.189	0.2	0.18	0.22
RacketSports	0.704	0.776	0.809	0.783
SelfRegulationSCP1	0.816	0.819	0.846	0.816
SelfRegulationSCP2	0.561	0.528	0.539	0.561
SpokenArabicDigits	0.927	0.925	0.929	0.957
StandWalkJump	0.467	0.667	0.333	0.467
UWaveGestureLibrary	0.828	0.85	0.875	0.903