Topological Data Analysis of Time-Series as an Input Embedding for Deep Learning Models

Morgan Byers[1][0000−0002−9854−8386], Lee B. Hinkle[2][0000−0001−7346−6344], and Vangelis Metsis[2][0000−0002−7371−8887]

[1] Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA morgan.byers@colorado.edu

[2] Department of Computer Science,Texas State University, San Marcos, TX 78666, USA
{leebhinkle,vmetsis}@txstate.edu

Abstract. An appreciable portion of medical data exist in the form of a time-series, e.g. electroencephalogram (EEG) and electrocardiogram (ECG) readings of the biopotentials related to the head and heart. Scarcity of labeled data and class imbalance pose challenges when training deep learning models. Topological data analysis (TDA) is an emerging area of research that can be applied to time-series data. In this paper we show that using TDA as a time-series embedding methodology for input to deep learning models offers advantages compared to direct training of such models on the raw data. In our work TDA acts as a generic, low-level feature extractor that is able to capture common signal patterns and thus improve performance with limited training data. Our experimental results on publicly available human physiological biosignal datasets show an improvement in accuracy, especially for imbalanced classes with only a few training instances compared to the full dataset.

Keywords: Topological Data Analysis · Time-Series Data · Embedding · Physiological Signals.

1 Introduction

Medical data is very often represented as a time-series, e.g. when recording the progression of symptoms over time. Additionally, when data is collected for medical research it is not uncommon to have a deficit of instances of rare conditions or scenarios. For example, in the case of human activity recognition (HAR), datasets may be largely comprised of instances of activities with a long duration such as walking and have only a few instances of events such as falling. As a result, those interested in researching the applications of machine learning to time-series medical data must work to overcome class imbalance that can interfere with training.

Topological data analysis (TDA) is an nascent area of interest to the data science community. Pesistent homology is a tool used by TDA that has recently been applied to time-series analysis with much success [7]. Since TDA is still in

its inception there is much room for exploring the ways this tool can be applied to deep learning. In this paper, we provide a data pipeline that takes advantage of the powerful tools of TDA in order to improve the per-class precision, recall and f1-scores of time-series data with underrepresented classes.

We use TDA as an embedding method to transform the raw time-series data before input to the deep learning model for training or prediction. The hypothesis is that similar signal patterns formed by time-series data will form similar topological representations, which can be learned more easily by deep learning models. The idea of input embedding is not new to the deep learning research community. For example, word embeddings are a common way of representing text as a vector for input to LSTM and Transformer models [13, 4]. Similar approaches have been used for image embeddings [26]. Although the general idea of time-series embeddings has been introduced in a few previous research works [14, 22], the use of TDA as an input embedding for deep learning models has so far been explored minimally. One example is the impact of TDA coupled with machine learning and the benefits for chaotic time-series data [24]. The deep learning pipeline described in this paper not only provides another tool to improve prediction scores for imbalanced data, but also demonstrates another application of the robust tools provided by TDA.

The rest of this paper is organized as follows. Section 2 provides background information on embeddings and topological data analysis. Section 3 describes the datasets and libraries used, as well as the overall methodology. Section 4 summarizes the results of our experiments. We conclude and suggest future work in Section 5.

2 Background

2.1 Time-series embeddings

Embeddings improve a machine learning algorithm's ability to model high-dimensional inputs such as sparse vectors representing sequential data in the form of time-series. A sequence of time-series data represents a set of measurements over time. Each data point (i.e. time-step) in the sequence carries very little information on its own, and it only creates a meaningful pattern when associated with multiple data points from neighboring time-steps. Furthermore, even for the same types of events, no two sequences are exactly the same, and added artifacts, such as measurement noise, push the data points farther apart in the vector space.

When large amounts of training data are available, as is usually the case with image datasets, deep neural networks perform well by learning the low-level features in the first few layers of the network and then combine the low-level features in deeper layers to form more complex patterns. However, in the medical domain the size of the available datasets are often not large enough to allow a deep neural network to be trained effectively. In such applications a more shallow network can benefit from an embedding layer which can capture

similarities between signal patterns. Ideally, an embedding captures some of the semantics of the input by placing semantically similar inputs close together in the embedding space. The wave2vec [27] library is a notable previous effort toward the creation of time-series embeddings.

2.2 Takens' Embedding

Performing TDA requires data in the form of a multidimensional time-series, so we must first perform an embedding to bring our data into higher dimensions. A Takens' embedding, also referred to as a time-delay embedding, is a method for embedding a time-series $x \in \mathbb{R}$ into a higher dimensional space. Performing such an embedding necessitates the choice of two parameters:

1. The window size, which will become the embedding dimension
2. The stride, which specifies how far along we move the window at each step

Choosing a stride that is too small will result in data that is highly overlapped and prone to over-fitting. Conversely, a stride which it too large with little to no overlap will result in a training set that may lose critical information near the window boundary. Thus, the choice of window size and stride must be tuned to each data set individually, as proper values for these parameters are crucial. Takens' theorem [19] explains that this embedding is topologically significant, given the correct choice of embedding parameters.

2.3 Topological Data Analysis (TDA)

The principle idea behind TDA is that discrete data are samples of an underlying continuous shape [7]. If we can properly reconstruct the features of this underlying continuous shape, then we can leverage those features as input for our deep learning models. Persistent homology is the field of mathematics that provides us with the very tools needed to perform this reconstruction. The particular mathematical feature that we look for with persistent homology is the number of holes present in our data. Because this idea is robust to outliers, it is very attractive for use in situations where data are particularly noisy [5].

The groundwork for our pipeline was laid by prior research in the field of TDA. Many applications of TDA to both time-series [7, 23] and other types of data [5, 8] have been explored. The use of Takens' embedding [19] for time-series data was shown to be effective in preparing data for TDA [18, 23]. However, much of what has been discovered with regards to TDA has been applied to traditional machine learning methods [15, 6]. One notable work is the previously mentioned application of TDA to volatile time-series data as input to a convolutional neural network [24].

2.4 Persistence Diagrams

After using a delay embedding on a time-series, we are left with a collection of vectors on which we can employ TDA. We do this by constructing simplicial

complexes from our data. A simplicial complex is a collection of simplices, which are the generalized definition of a triangle. Simplices exist in all dimensions; a 0-simplex is a singular point, a 1-simplex consists of two points joined by a line segment, and a 3-simplex is what we know as a traditional triangle. More generally, a k-simplex consists of a collection of k+1 vertices residing in $\mathbb{R}^{k+1}$ [6].

As mentioned above, we begin with each instance consisting of a discrete set of vectors in $\mathbb{R}^d$. This is sometimes referred to as a data cloud. With this data cloud, we can construct a simplicial complex. Specifically, we create a Vietoris-Rips complex, often abbreviated as a Rips complex. The act of creating this complex is known as Rips filtration. More information about the construction of these complexes can be found in [8].

Of interest to us is the number of holes, called a Betti number, that appear in our data as we perform this Rips filtration. Formally, the i^{th} Betti number, β_i is the rank of the i^{th} homology group of a topological space. Informally, the i^{th} Betti number is the number of (i+1)-dimensional holes in our data. We are primarily interested in the number of loops, or 2-dimensional holes, β_1, present in our data cloud.

However, the Betti numbers of a single Rips complex constructed in isolation reveals little about the data. We must determine which of these holes persist in the Rips complexes that are constructed at different values of ϵ [8]. This requires the construction of a persistence diagram.

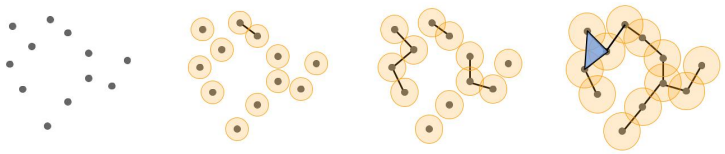


Fig. 1: An example of Rips filtration.

A persistence diagram is a graph that provides information about the components and loops that appear in the data as ϵ increases in value. Imagine a set of discrete points that constitute an incomplete set of connect-the-dots. Rather than playing the game traditionally, where the dots would be connected according to a predefined order, a different approach is used. This begins with drawing an arbitrarily small circle around each point in the connect-the-dot game. Incrementally larger and larger circles are drawn around each point until two or more of those circles intersect. When two dots' circles intersect, they are connected. As more and more dots are connected, loops will start to appear (and eventually disappear) in the data and individual components will start to disappear. For each component or loop, a point (x, y) is plotted in the persistence diagram so that x represents the radius of the circles when the component or loop first appeared (called its "birth") and y represents the radius of the circles when the

component or loop disappeared (its “death"). The persistence diagram gives us insight into the true structure of our data set - the loops and holes which persist throughout many values of ϵ are true features of our data, which the machine learning models can then utilize when making predictions.

After constructing a persistence diagram, it is likely that many points in the diagram will be clustered around the line $y = x$. The points that are further away from the line $y = x$ are considered persistent. An example of one of our persistence diagrams is shown in Figure 2.

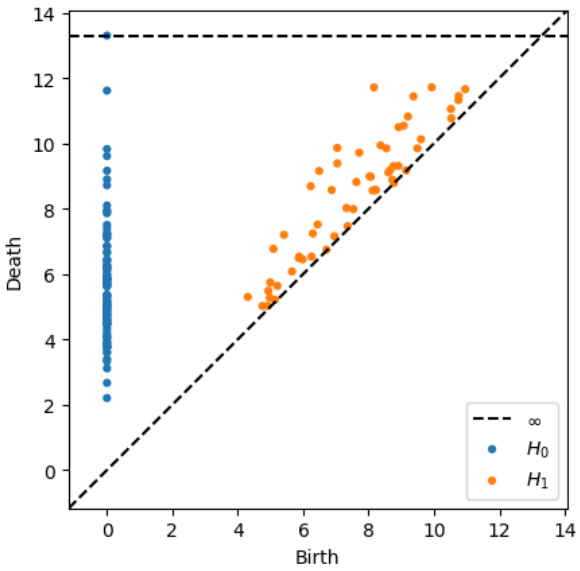


Fig. 2: An example of a persistence diagram. Note that ‘Birth’ refers to the radius at which the feature appeared and ‘Death’ refers to the radius at which the feature disappeared. New components are not created after a radius of 0. Instead, when two componenets’ balls intersect, one component dies and the other lives on.

Although persistence diagrams are powerful, they are not suitable as input for many popular machine learning algorithms. This is where a persistence image becomes useful. Persistence images offer a stable, vector-based representation of persistence diagrams that work as input for many machine learning algorithms. As described in [1], to create a persistence image, we begin by mapping a persistence diagram PD to an integrable function $\rho_{PD} : \mathbb{R}^2 \to \mathbb{R}$ that’s defined as a weighted sum of probability density functions (one centered at each point in PD). Then a grid is defined by taking a discretization of a subdomain of ρ_{PD}. Finally, a persistence image is yielded by taking an integral of the function on each grid box (Fig. 3).

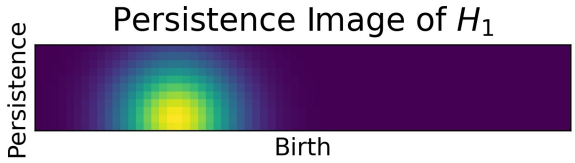


Fig. 3: An example of a persistence image

3 Methodology

3.1 Datasets

Multiple datasets were used in this work, including three human activity recognition (HAR) datasets and one Electroencephalogram (EEG) dataset. The UniMiB SHAR [12] dataset contains acceleration data captured using a smartphone on 30 subjects and includes nine types of activities and seven types of falls. The MobiAct [25] dataset includes both accelerometer and gyroscope (rotation) data also recorded on a smartphone with 50 subjects. It includes nine activities and four types of falls. The third activity dataset is UCI HAR [2] which contains smartphone accelerometer and gyroscope data for 30 subjects performing six physical activities. The final dataset used was the EEG Motor Movement/Imagery [9, 17] dataset. For this dataset we worked with Task 3.

3.2 Tools and Libraries

The models were implemented using Python. The NumPy [10] and Sci-kit Learn [16] libraries were used to preparing the data and the deep learning models were built using Tensorflow Keras [3]. The Takens' Embedding was performed with the Giotto-tda library [20]. We used Ripser [21] and Persim [1] to perform the topological data analysis.

3.3 Data Pipeline

Since applying TDA necessitates that each instance consists of a collection of vectors, in the case of a single channel time-series, we must first embed the data from $\mathbb{R}$ into $\mathbb{R}^d$, where $d \in \mathbb{N}$ is the embedding dimension, by using a Takens' embedding with a stride of one. In the case of a multi-channel time-series, we consider each variable to be already embedded in $\mathbb{R}^t$, where t is the number of time steps. For each of the human activity recognition data sets the quadratic mean (RMS) of the acceleration in the x, y, and z axes is used as a single channel time-series. Takens' embedding is performed on the data before continuing with the pipeline. In the case of EEG classification we do not use Takens' embedding and instead use each channel of the EEG data as-is, because our EEG data is already highly dimensional (See Fig. 4). After the data has been prepared, a

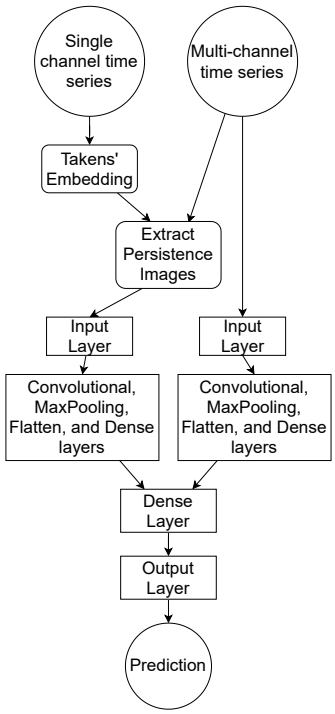


Fig. 4: The setup of our data pipeline and our baseline comparison.

baseline deep learning estimator is trained solely on the data without performing topological data analysis. The architecture of our baseline model consists of the same layers that the non-TDA side of our final model uses (see Fig. 4), followed by additional dense layers for output prediction.

In the case of single channel data, the baseline classifier is trained on the data after Takens' embedding has been applied. The architecture of this baseline estimator is variable and can be changed to suit the particulars of the specific data type. For instance, when creating the baseline model for classifying EEG data, the model described in [11] was used. After the initial model has been trained, the probabilities output by the model's predictions are used with the test set for evaluation.

At the same time, we use the embedded data to extract topological features from our time-series. A Rips filtration [21] yields persistence diagrams (Fig. 2) from each instance. Persim [1] is used to convert the persistence diagrams into persistence images, which are stable vector representations of persistence diagrams. These persistence images will be used as input to our final model. The particular pixel size and dimensions of each persistence image are hyperparameters that are tuned to fit each time-series individually. In our case, these hyper-

parameters were determined by using five-fold cross-validation of the training set with the final pipeline.

The persistence images are used as input data to one side of our final model (see our pipeline in Fig. 4). The basic architecture of this side of our model will remain the same for any data, and consists of 1D convolutional layers, dropout, maxPooling, flatten, and dense layers. However, hyperparameters like kernel and batch size must be tuned to achieve the best performance on each set of data. The deep learning model is completed by concatenating the outputs of last layers of our two networks: the network trained on traditional features and the network trained on the topological features. This is passed through a dense layer and then to a final output layer.

To evaluate our pipeline, the classification reports yielded by both the baseline and final models were compared. Group based five-fold cross validation was used to evaluate the models' performance. Since all of our datasets are segmented by subject, we delineated each fold by subject. For instance, all of subject one's data would be included in the test set for one fold, and then subject one's data would be in the training set for the four remaining folds. In the case of the UCI HAR data set, the data is already segmented into a training and testing sets. As a result, in order to avoid overfitting, we perform only one trial with this model, training on the provided training set and evaluating on the provided test set.

4 Results

In Tables 2 through 5, results are formatted similarly to a classification report. In these tables, 'Base' refers to the metrics obtained by our baseline classifier, while 'TDA' refers to the metrics obtained by our final deep learning model. Our TDA deep learning approach yielded a statistically significant improvement over the baseline in almost every case of data that we tried (Table 1). The best results were obtained on the MobiAct dataset, which is perhaps because of the severity of class imbalance present in the data set. In fact, just two classes comprise roughly 70% of instances in the data set. Although the f1-score has improved for

Table 1: Hypothesis Test for Difference in Mean Recall and f1-Score ($H_0 : \mu_d \leq 0$)

DataSet	p-value (recall)	p-value (f1-score)
MobiAct	0.013	0.009
UniMiB	0.018	< 0.001
UCI HAR	0.336	0.051
EEG	0.255	0.025

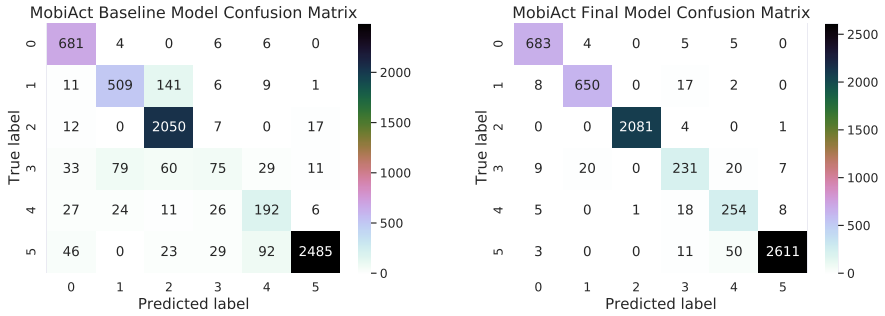
a majority of the classes in each data set, due to the imbalanced nature of most of these data sets, the overall prediction accuracy rarely changed by a significant amount. As mentioned previously, the most significant improvement in accuracy

Table 2: MobiAct Classification Report

Class	Precision		Recall		f1-score		Support
	Base	TDA	Base	TDA	Base	TDA	
0	0.84	**0.96**	0.98	0.98	0.90	**0.97**	697
1	0.83	**0.96**	0.75	**0.96**	0.79	**0.96**	677
2	0.90	**1.00**	0.98	**1.00**	0.94	**1.00**	2086
3	0.50	**0.81**	0.26	**0.80**	0.34	**0.81**	287
4	0.59	**0.77**	0.67	**0.89**	0.63	**0.82**	286
5	0.99	0.99	0.93	**0.98**	0.96	**0.98**	2675

came from the MobiAct data. In particular, our TDA pipeline improved the f1-score of every class and improved the recall of five of the six classes (Table 2).

The TDA pipeline also showed an increase in recall and f1-score for the UniMiB data. The per-class f1-score improved in each of the nine classes, and eight classes also showed an improvement in recall.



(a) Baseline classifier (b) TDA-based model

Fig. 5: Confusion matrices comparing the performance of the baseline classifier against our TDA-based model.

Similarly, on the UCI HAR data our pipeline improved the f1-score for five of the six classes; the overall classification accuracy increased from 58% to 60%.

The EEG dataset consists of only two classes. In this case our pipeline improved the f1-score for both classes; the overall accuracy increased by 2%. Typically, the classes that showed improvement in f1-score consisted of fewer instances than that of the classes which showed no improvement.

5 Conclusion

The experiments detailed in this paper provide a novel deep neural network embedding method utilizing Topological Data Analysis. The presented data pipeline

Table 3: UniMiB Classification Report

Class	Precision		Recall		f1-score		Support
	Base	TDA	Base	TDA	Base	TDA	
0	0.73	**0.80**	0.63	0.63	0.68	**0.70**	153
1	0.53	**0.54**	0.37	**0.51**	0.44	**0.53**	216
2	0.89	**0.93**	**0.90**	0.89	0.90	**0.91**	1738
3	0.96	0.96	0.94	**0.97**	0.95	**0.96**	1985
4	**0.85**	0.83	0.77	**0.82**	0.81	**0.82**	921
5	0.90	**0.96**	0.95	**0.96**	0.93	**0.96**	746
6	0.79	**0.83**	0.88	0.88	0.83	**0.85**	1324
7	0.52	**0.54**	0.55	**0.60**	0.53	**0.57**	296
8	0.64	**0.72**	0.58	**0.67**	0.61	**0.69**	200

Table 4: UCI HAR Classification Report

Class	Precision		Recall		f1-score		Support
	Base	TDA	Base	TDA	Base	TDA	
0	0.75	**0.82**	**0.90**	0.88	0.82	**0.85**	496
1	**0.84**	0.81	0.60	**0.75**	0.70	**0.78**	471
2	0.83	**0.87**	**0.90**	0.87	0.86	**0.87**	420
3	0.35	**0.37**	**0.50**	0.37	**0.41**	0.37	491
4	**0.40**	0.38	0.36	**0.38**	0.37	**0.38**	532
5	**0.42**	0.41	0.30	**0.41**	0.35	**0.41**	537

Table 5: EEG Classification Report

Class	Precision		Recall		f1-score		Support
	Base	TDA	Base	TDA	Base	TDA	
0	0.58	**0.61**	**0.59**	0.58	0.59	0.59	445
1	0.59	**0.61**	0.58	**0.63**	0.59	**0.62**	455

and results show increased per-class recall and f1-scores for underrepresented classes that may be applicable to a variety of time-series data. Since medical research is typically interested in identifying rare occurrences (e.g. seizure detection), it is expected that much medical data is imbalanced, with many more normal instances than those experiencing an anomaly. Finding a methodology to represent persistence diagrams in such a way that transformers are capable of learning from these representations remains an open question.

Acknowledgements

This material is based upon work supported by the National Science Foundation under REU Site Grant No. 1757893. The research work was conducted while Morgan Byers was an undergraduate student at Texas State University. Any opinions, findings, and conclusions or recommendations expressed in this

material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Hanson, E., Motta, F., Ziegelmeier, L.: Persistence images: A stable vector representation of persistent homology. Journal of Machine Learning Research **18**, 1–35 (2017), http://jmlr.org/papers/v18/16-337.html.
2. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.: A public domain dataset for human activity recognition using smartphones. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (April 2013)
3. Chollet, F., et al.: Keras. https://keras.io (2015)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
5. Edelsbrunner, H., Harer, J., et al.: Persistent homology-a survey. Contemporary mathematics **453**, 257–282 (2008)
6. Gholizadeh, S., Seyeditabari, A., Zadrozny, W.: Topological signature of 19th century novelists: Persistent homology in text mining. Big Data and Cognitive Computing **2**, 1–10 (12 2018). https://doi.org/10.3390/bdcc2040033
7. Gholizadeh, S., Zadrozny, W.: A short survey of topological data analysis in time series and systems analysis. arXiv (10 2018), http://arxiv.org/abs/1809.10745
8. Ghrist, R.: Barcodes: the persistent topology of data. Bulletin of the American Mathematical Society **45**(1), 61–75 (2008)
9. Goldberger, A.L., Amaral, L.A., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E.: Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. Circulation **101** (2000). https://doi.org/10.1161/01.cir.101.23.e215, https://pubmed.ncbi.nlm.nih.gov/10851218/
10. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2
11. Lawhern, V.J., Solon, A.J., Waytowich, N.R., Gordon, S.M., Hung, C.P., Lance, B.J.: Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. Journal of Neural Engineering **15**, 056013 (10 2018). https://doi.org/10.1088/1741-2552/aace8c, https://iopscience.iop.org/article/10.1088/1741-2552/aace8c
12. Micucci, D., Mobilio, M., Napoletano, P.: Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. Applied Sciences **7**(10) (2017). https://doi.org/10.3390/app7101101, http://www.mdpi.com/2076-3417/7/10/1101
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)

14. Nalmpantis, C., Vrakas, D.: Signal2vec: time series embedding representation. In: International Conference on Engineering Applications of Neural Networks. pp. 80–90. Springer (2019)
15. Obayashi, I., Hiraoka, Y.: Persistence diagrams with linear machine learning models. arXiv (6 2017), http://arxiv.org/abs/1706.10082
16. 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., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
17. Schalk, G., McFarland, D.J., Hinterberger, T., Birbaumer, N., Wolpaw, J.R.: Bci2000: A general-purpose brain-computer interface (bci) system. IEEE Transactions on Biomedical Engineering **51**, 1034–1043 (6 2004). https://doi.org/10.1109/TBME.2004.827072, https://pubmed.ncbi.nlm.nih.gov/15188875/
18. Seversky, L.M., Davis, S., Berger, M.: On time-series topological data analysis: New data and opportunities. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 59–67 (2016)
19. Takens, F.: Detecting strange attractors in turbulence. In: Dynamical systems and turbulence, Warwick 1980, pp. 366–381. Springer (1981)
20. Tauzin, G., Lupo, U., Tunstall, L., Pérez, J.B., Caorsi, M., Medina-Mardones, A., Dassatti, A., Hess, K.: giotto-tda: A topological data analysis toolkit for machine learning and data exploration (2020)
21. Tralie, C., Saul, N., Bar-On, R.: Ripser.py: A lean persistent homology library for python. The Journal of Open Source Software **3**(29), 925 (Sep 2018). https://doi.org/10.21105/joss.00925, https://doi.org/10.21105/joss.00925
22. Tran, Q.H., Hasegawa, Y.: Topological time-series analysis with delay-variant embedding. Physical Review E **99**(3), 032209 (2019)
23. Ty, A.J., Fang, Z., Gonzalez, R.A., Rozdeba, P.J., Abarbanel, H.D.: Machine learning of time series using time-delay embedding and precision annealing. Neural Computation **31**(10), 2004–2024 (2019)
24. Umeda, Y.: Time series classification via topological data analysis. Information and Media Technologies **12**, 228–239 (2017). https://doi.org/10.11185/imt.12.228
25. Vavoulas, G., Chatzaki, C., Malliotakis, T., Pediaditis, M., Tsiknakis, M.: The mobiact dataset: Recognition of activities of daily living using smartphones. In: International Conference on Information and Communication Technologies for Ageing Well and e-Health. vol. 2, pp. 143–151. SciTePress (2016)
26. Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., Wu, Y.: Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1386–1393 (2014)
27. Yuan, Y., Xun, G., Suo, Q., Jia, K., Zhang, A.: Wave2vec: Learning deep representations for biosignals. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 1159–1164. IEEE (2017)