Enhancing Convolutional Neural Networks with Topological Data Analysis

Franciszek Sobota
e-mail: [1] sobotafran@gmail.com,

August 5, 2025

1 INTRODUCTION

First, I think I need to explain myself regarding this catchy title. In this simple half article, half tutorial, I will try to present paper *Detection of gravitational waves using topological data analysis and convolutional neural networks* [1]. The authors are trying to determine whether or not signals contain a gravitational wave footprint. This is quite challenging due to the high level of noise in the signal. They improved the earlier approach that was applying a CNN to the bare signal to classify them by providing additional features using topological data analysis (TDA). We will follow their TDA pipeline and use and explain time series *sliding window* embeddings and persistent homology to obtain significantly better performance of the classifier.

2 DATA

Before we dive into analysis, we need to prepare and understand the data. The signals will be artificially generated by the surrogate model to create reference signals with different mass ratios of the colliding black holes. Finally, we add the signals to the Gaussian noise with standard deviation 1, as follows:

$$s = g + \epsilon \frac{1}{R} \xi$$

g is the signal from the reference set, R is the constant corresponding to the Signal to Noise Ratio (SNR), ξ is the noise mentioned above. Lastly, the $\epsilon = 10^{-19}$ is a scaling constant. In this way, we are able to generate arbitrarily many training samples for our model. One can see an example of such generated signals in Figure 1, where both background noise alone and background noise with added signal are plotted. Using only the plots, it is not as obvious which one from the plots is the one with the signal, especially when you don't know where signal is located. Moreover, I have used the maximum SNR $= 17.98$ among those used in the article [1] for the purposes of this text.

3 TAKENS' THEOREM

Now when data is prepared, we can go to our TDA pipeline. As I have mentioned in the introduction, we are going to use persistent homology to obtain some features that we are going to interpret. But first we have to have some space on which we can compute homology. Typically in TDA we are dealing with point clouds that then are transformed into filtration space using, for example, Rips Complexes. In this case, it will be no different, but we need to transform our time series into a point cloud first. The theorem, referenced in the section title, about reconstruction will provide us motivation for such an approach.

I will first provide the informal visual intuition for why this approach may work, and then I will proceed to formulate the Takens' Theorem.

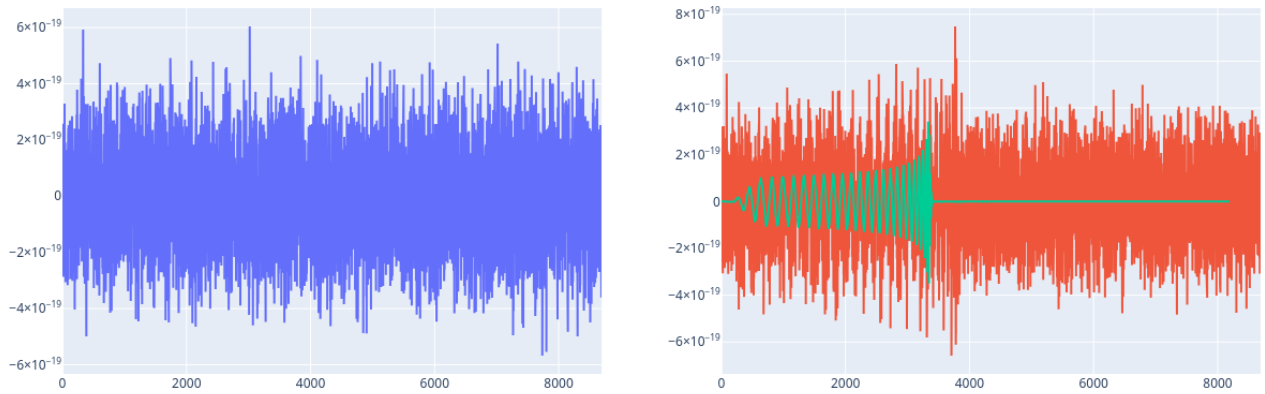

Figure 1: On the left plot (blue), there is random background noise without a signal. On the right (red), background noise with signal added (green trace, bare signal).

3.1 The intuition

Okay, I have told much about the high-level overview, so it is time to explain how to construct the point cloud. Let us consider a time series $X = \{x_n\}$ and $w \in \mathbb{N}$ window size. Now our point cloud would be $\{(x_n, x_{n+1}, \ldots, x_{n+w-1})\} \subset \mathbb{R}^w$. The method is called sliding window embedding. Fair and straightforward enough. Now let me present a simple but illustrative example.

3.1.1 Henon system

The Henon system is a dynamical system that is often presented as an example in this case (for instance, in [2]). I am also going to cite it. So the system itself is defined by the mapping:

$$\begin{cases} x_{n+1} = 1 - ax_n^2 + by_n \\ y_{n+1} = x_n \end{cases} \tag{3.1}$$

The time series that we are going to focus on is $\{x_n\}$. I have generated this sequence with parameters as in [2]. I have also provided a shuffled version of it. This can be viewed as a random i.i.d. sequence with the same histogram. One can see both series in Figure 2. Both plots look rather chaotic and random. What if we apply the *sliding window*?

In Figure 3, one can see the result of applying the sliding window embedding to the previous time series. The shuffled one is just the random point cloud, while the one from the non-shuffled Henon series forms the well-known Henon attractor shape, a set of points in the phase space of the Henon map to which trajectories tend after many interations. Is this a coincidence? Maybe, but maybe not. In a moment it will turn out that, under certain conditions, we could indeed expect that if the time series came from a dynamical system, the *sliding window* embedding would include some useful features.

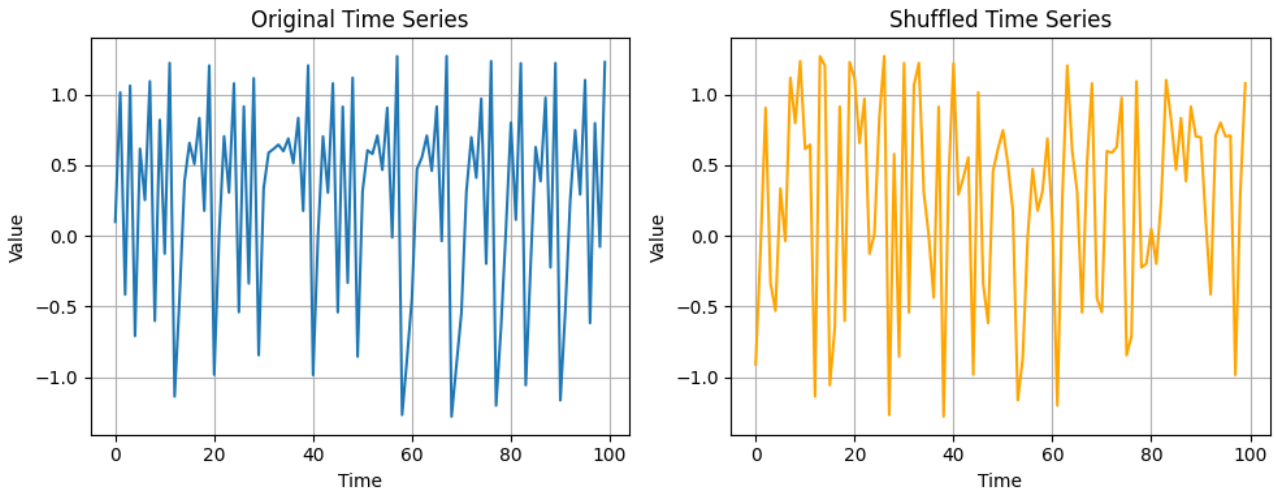

Figure 2: Time series for the artificially generated Henon system (projected on the first coefficient) on the left, and the shuffled version on the right.

4 Theorem

Now, when we already have intuition why *sliding window* may work, let's state Takens' Theorem. This claim was presented and proven by, unsurprisingly, Floris Takens' in [3]. I will not present the proof that is quite technical. Instead I will rely on intuition. Below, there is the formulation of the theorem from [2].

Theorem 1 (Takens' theorem) *For a compact m-dimensional manifold M, $l \geq 1$, and $k \geq 2m$ there is an open and dense subset $\mathcal{U} \subset Diff^l(M) \times C^l(M)$, the product of the space of C^l - diffeomorphisms on M and the space of C^l-functions of M, such that for $(\varphi, f) \in \mathcal{U}$ the following map is an embedding of M into $\mathbb{R}^k$:*

$$M \ni x \mapsto \Big(f(x), f(\varphi(x)), \ldots, f(\varphi^{k-1}(x))\Big) \in \mathbb{R}^k \tag{4.1}$$

This may seem quite complicated at first, but I will try to connect this with the previous example. So M from the theorem would be, loosely speaking, the Henon attractor, a subspace of $\mathbb{R}^2$. Why 'loosely'? Among others, because our points tend to the attractor but are not exactly from it. Next, let $\varphi \in \text{Diff}^l(M)$, and set $k = 2$. φ would be a transition map from equation 3.1, and lastly $f(x_n, y_n) = x_n$ is an observation map that maps the full state into a observable scalar. Now we would expect that if we are lucky and $(\varphi, f) \in \mathcal{U}$, the map in equation 4.1 is an embedding, and we will be able to see, possibly deformed, the shape of Henon attractor on the scatter plot of the *sliding window* embedding.

This is indeed the case. If you think about it, it is not particularly surprising that the attractor occurs. After all, the transition for y_n was $y_{n+1} = x_n$. What's more, this example is not exactly strict. Some assumptions are not exactly satisfied. Despite this, it illustrates the idea well, so I decided to use it.

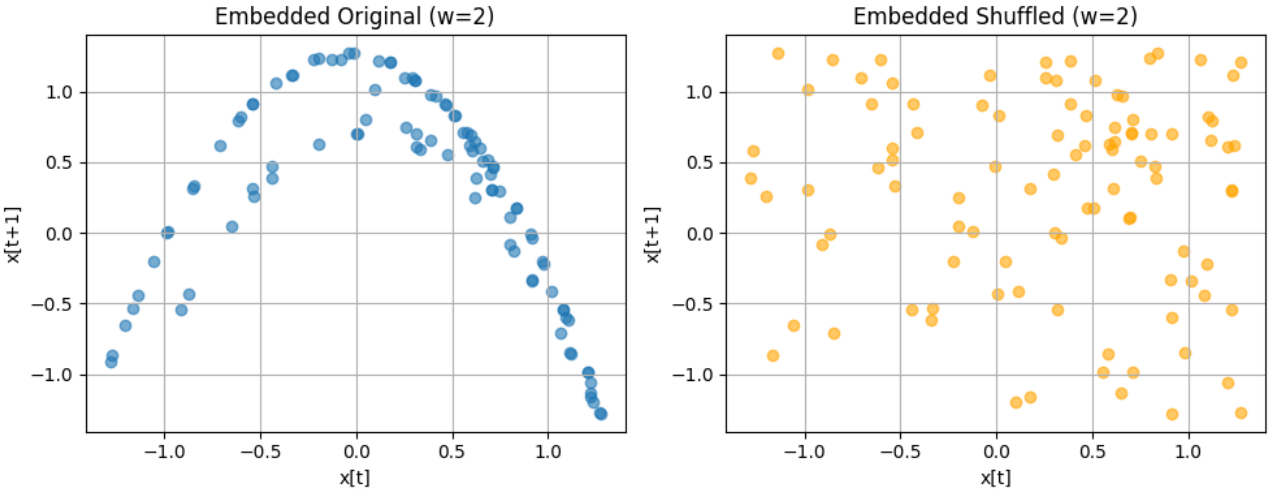

Figure 3: On the left is the Henon-derived signal embedded in $\mathbb{R}^2$, and on the right is the shuffled version.

4.1 Gravitational waves

Okay, it is high time to come back to our gravitational waves. We already know how to embed time series into point clouds. Let's then embed our signals into $\mathbb{R}^{200}$ first and then apply PCA to reduce it to 3 dimensions, and take a closer look (Figure 4).

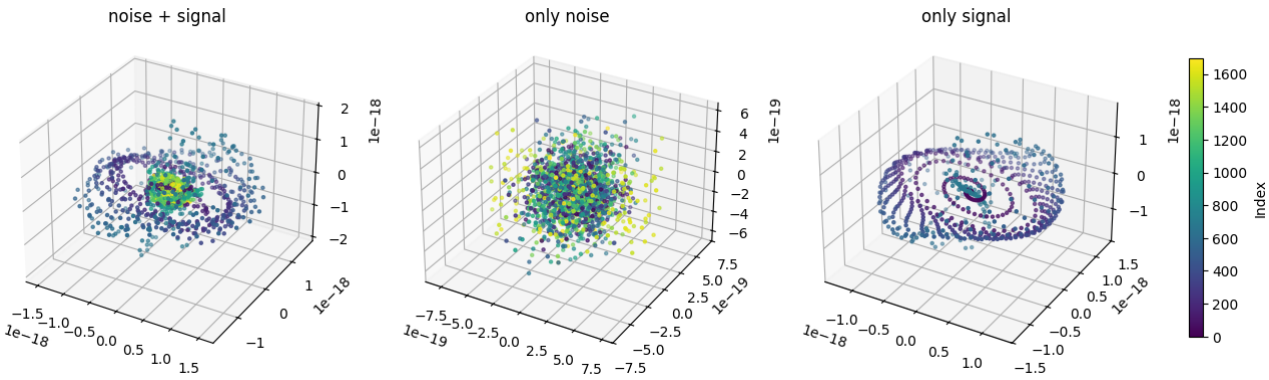

Figure 4: Clouds obtained with *sliding window* method, after PCA reduction to 3 dimensions

Now we can clearly see the difference! The only noise signal results in the random point cloud. The bare signal results in the nice and spiral-structured one, while the mixed signal is something in between. We already can see the difference, but we would also want this distinction to be detectable computationally.

One way to describe the "only signal" point cloud is to say that the point cloud obtained from the noise with signal contains some loop-like shapes. This is another important feature of the *sliding window* embedding. Periodic function embeddings result in one-dimensional closed traces in the space. So one may look at the level at which it resembles a curve as the level of how periodic the function is. Fortunately, persistent homology is a tool that is able to find those and "measure", among others, how loop-y the point cloud is.

5 PERSISTENT HOMOLOGY

Persistent homology is the most fundamental technique in the TDA, it allows us to detect some patterns and structures in point clouds, such as, for example, loops that we want to find.

5.1 Homology

To do this, first describe how to count n-dimensional holes in some simplicial complex X. The simplicial complex is the set of points and simplices on them. To calculate holes in such space, we are going to define what's called the simplicial homology group $H_n(X)$. The number $\beta_n = \dim H_n(X)$, called the n-th Betti number, would be roughly the number of n-dimensional holes in the space X. For example H_0 counts connected components, while H_1 captures loops.

For our purposes, let's fix ring $R = \mathbb{Z}_2$, but in general it doesn't need to be a field. The *module of simplicial k-chains* would be defined as:

$$C_k(X, R) = R^{|X_k|}$$

where, X_k is set of k-dimensional simplices in X. Next let's fix the order on the vertices and define so-called boundary map:

$$\partial_k : C_k(X, R) \to C_{k-1}(X, R)$$

$$\partial_k([v_0, v_1, \ldots, v_n]) = \sum_i (-1)^i [v_0, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n]$$

Finally we can define the k-th homology group as:

$$H_k(X, R) = {}^{ker(\partial_k)}\!/_{im(\partial_{k+1})}$$

The definition may seem complicated, but I found the intuition about the boundary operator when $k = 1$ very helpful. One can think that $ker(\partial_1)$ contains closed paths on the simplexes (those without boundary i.e. cycles), while $im(\partial_2)$ are those closed paths that are boundaries of higher-dimensional simplexes (the two-dimensional i.e. filled triangles) and thus are not real holes, only filled cycles.

5.2 Persistent

But we only have a point cloud, not a simplicial complex. How can we get one to compute the simplicial homology of it? There are many ways to do it. One of the easiest is to get the Rips complex. The simplex $[v_0, \ldots, v_n]$ is included in the complex if and only if $\forall i, j \; d(v_i, v_j) < \varepsilon$ for some chosen metric d. We will stick with the Euclidean one. Okay, one may say, but how do we choose this parameter ε so that we will be able to find exactly those loops that we see in the plot? And now the best part, we do not choose. We are calculating the homology group for increasing values of ε until we get the full simplicial complex, with every simplex included. Then those relatively larger structures, for example holes, should *persist* across a range of ε values, and we will be able to detect them.

There are many ways to represent persistent homology as a feature and feed it into such a model as a neural network or any other statistical reasoning. In the original paper [1], they used so-called *persistent vectors*. In this tutorial [4], which helped me a lot during the implementation, they decided to use *persistence entropy*. I

decided to test another representation method and chose the *Betti curve* for 0-th and 1-st dimension, which is just the sequence of Betti numbers for different epsilons. It encodes the number of connected components and loops for varying scales.

6 NEURAL NETWORK AND RESULTS

I have implemented the whole pipeline using Python and the GUDHI library for the TDA [5]. I have also added a simple neural network, inspired by the one used by the authors of [1]. I have created two convolutional backbones. The first one was responsible for processing the raw noisy signals and was constructed exactly the same way as in the article. The second backbone was responsible for preprocessing the Betti curve of the signal and was significantly smaller. Then the outputs of the backbones are concatenated and transferred to the classification head.

I have trained two networks. One used both the Betti curve and the raw signal as input features, while the second one used only the raw signal. Both nets were trained with 40 epochs with the 900 samples with $R = 0.65$ and tested with 100 samples. As an optimizer, I have used ADAM with default PyTorch parameters.

This text is supposed to be about TDA, not neural networks, so I will skip reporting the fancy metrics such as ROC and stick to accuracy. The network with the Betti curve provided as a feature had 99% accuracy on the test set, while the network without it showed no prediction ability, reaching a dizzying 51% accuracy on the test set.

Also because of that I did not attempt to fine-tune the models to get the best possible performance. My goal was to prove that the TDA-derived features can be extremely helpful in such tasks, and I believe I have successfully demonstrated that.

All the code used to obtain the plots and train the model is available on github.

REFERENCES

[1] C. Bresten and J.-H. Jung, *Detection of gravitational waves using topological data analysis and convolutional neural network: An improved approach*, 2019. arXiv: 1910.08245 [astro-ph.IM]. [Online]. Available: https://arxiv.org/abs/1910.08245.

[2] H. Broer and F. Takens, "Reconstruction and time series analysis," in *Dynamical Systems and Chaos*. New York, NY: Springer New York, 2011, pp. 205–242, ISBN: 978-1-4419-6870-8. DOI: 10.1007/978-1-4419-6870-8_6.

[3] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*, D. Rand and L.-S. Young, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 366–381, ISBN: 978-3-540-38945-3.

[4] *Gravitational wave detection*. [Online]. Available: https://giotto-ai.github.io/gtda-docs/latest/notebooks/gravitational_waves_detection.html.

[5] T. G. Project, *GUDHI*, 3.11.0. GUDHI Editorial Board, 2025. [Online]. Available: https://gudhi.inria.fr/doc/3.11.0/.