Predicting the failure of Component X in the Scania dataset with Graph Neural Networks

Maurizio Parton[1,7][0000−0003−4905−3544], Andrea Fois[2][0000−0002−2749−240X], Michelangelo Vegliò[1], Carlo Metta[3,6][0000−0002−9325−8232], and Marco Gregnanin[4,5][0009−0004−3205−6997]

[1] University of Chieti-Pescara, Italy
[2] University of Parma, Italy
[3] ISTI-CNR, Pisa, Italy
[4] IMT School for Advanced Studies Lucca, Italy
[5] KU Leuven, Belgium

Abstract. We use Graph Neural Networks on signature-augmented graphs derived from time series for Predictive Maintenance. With this technique, we propose a solution to the Intelligent Data Analysis Industrial Challenge 2024 on the newly released SCANIA Component X dataset. We describe an Exploratory Data Analysis and preprocessing of the dataset, proposing improvements for its description in the SCANIA paper.

Keywords: SCANIA Component X, Predictive Maintenance, Graph Neural Networks, Visibility Graphs

1 Introduction

The growing sophistication and accessibility of machine learning and deep learning models have significantly impacted the automotive industry, particularly in maintenance practices. Leveraging big data and data-driven models has revolutionized vehicle maintenance by enabling the implementation of Predictive Maintenance (PdM) strategies. PdM aims to optimize vehicle performance and prevent component failure, thereby minimizing maintenance costs. However, the scarcity of real-world PdM datasets presents a substantial challenge. Most existing datasets rely on synthetic data, which introduce biases into the models.
This paper addresses this challenge by investigating the performance of two models on a real-world multivariate timeseries dataset, "SCANIA Component X", for predicting maintenance needs in a vehicle fleet. One of the novelties introduced in this paper is the application of Graph Neural Networks (GNNs) [2] to PdM. GNNs demonstrate exceptional capabilities in processing data structured as graphs, making them well-suited to capture the complex and intricate

[6] EU Horizon 2020: G.A. 871042 SoBig-Data++, NextGenEU - PNRR-PEAI (M4C2, investment 1.3) FAIR and "SoBigData.it".
[7] Funded by GNSAGA INdAM group.
[all]Computational resources provided by CLAI laboratory, Chieti-Pescara, Italy.

relationships present within sensor data collected from vehicles. We introduce a framework that utilizes both time series signatures and visibility graphs for effective PdM analysis. Timeseries signatures, a concept derived from path theory, serve as a potent tool for identifying the most relevant variables crucial for constructing an informative graph representation of the multivariate timeseries data. Subsequently, these identified key variables are used to construct the graph using the visibility graph algorithm, which is able to transform a time series in a graph preserving its temporal characteristics such as trends [3, 10].
Section 2 details the methods. Sections 3 and 4 focuses on data analysis and preprocessing. Section 5 describe hyperparameters and results, and Section 6 provides potential avenues for future research.

2 Methods

Geometric Deep Learning. Graph Neural Networks (GNNs) represent a specific deep learning model tailored to data structured as a graph. When dealing with multivariate time series, determining the graph structure becomes imperative, as no predefined structure exists as in social networks, chemistry, or traffic forecasting problems [4]. Two primary approaches emerge: constructing a graph representation from a similarity matrix, typically utilizing the correlation matrix, or selecting a variable and computing the graph using the visibility graph method [3]. In our analysis, we opt for the latter approach due to its simplicity and effectiveness. Utilizing the correlation matrix necessitates filtering it using a threshold method to eliminate noise and irrelevant connections in the graph. Not filtering creates a complete graph with all nodes interconnected, making it complex and less useful.
Signature. The concept of the signature originates from the domain of path theory, offering a structured and comprehensive depiction of the temporal evolution within a time series. Its efficacy lies in capturing both temporal and geometric patterns inherent in the time series. Temporal patterns encompass long-term dependencies and recurrent trends over time, while geometric patterns involve the shape of time series trajectories, including intricate behaviors such as loops and self-intersections [6]. The signature structure offers a hierarchical interpretation, with lower-order components capturing broad path attributes and higher-order terms revealing intricate characteristics, including higher-order moments and local geometric features. Importantly, the signature remains invariant under reparameterization, preserving integral values despite time transformations, and adheres to translation invariance and concatenation properties [7].
Furthermore, when representing a stochastic process as a cumulative process and computing its signature, referred to as the "cumulative signature," we can effectively capture and portray temporal patterns. Specifically, the cumulative signature aggregates information over time, thereby encapsulating the temporal evolution of the data. Additionally, the final cumulative signature is influenced by the entire sequence of signatures, reflecting the order in which information accumulates over time. Lastly, it mitigates noise and fluctuations in the original

data, offering a more reliable representation of the underlying patterns [6].
Consider a discrete time series $S = \{s_0, s_1, \ldots, s_T\}$, and let $\mathrm{T}((\mathbb{R}^d))$ be the tensor algebra $\oplus_{k=0}^{\infty}(\mathbb{R}^d)^{\otimes k}$, encompassing the signatures of $\mathbb{R}^d$-valued paths. As the signature method is applicable solely to continuous processes, discrete time series must undergo a transformation into a continuous form. This transformation can be achieved through methods such as the lead-lag transformation or the time-join transformation [8]. We opt to focus on continuous functions mapping from a compact time interval $J := [a, b]$ to $\mathbb{R}^d$ with finite p-variation, denoted as $C_0^p(J, \mathbb{R}^d)$. Thus, we choose the lead-lag transformation, denoted by L, to convert the discrete time series S into a continuous form due to its ability to extract path volatility. Consequently, we define the signature, denoted as $\mathcal{S}$, and the truncated signature at level M, denoted as $\mathcal{S}_M$, as outlined in [9]. The rationale behind utilizing the signature of a time series to compare two or more stochastic processes stems from the "Expected Signature" theorem [12]. This theorem asserts that if the signatures computed on two distinct stochastic processes are equal, then the processes themselves are equal in distribution. The expected signature theorem enables the comparison of different stochastic processes because, under suitable assumptions, the expected signature uniquely determines the distribution of the stochastic process, analogous to the role of the moment generating function [13]. For a more comprehensive and precise mathematical definition, please refer to [6, 14, 15].

Visibility Graph In this study, we employ the visibility graph algorithm proposed in [3] to derive the graph representation. The visibility graph algorithm operates specifically on univariate time series data. To identify the time series suitable for graph representation, we opt to compute the truncated signature for each time series. Subsequently, leveraging the "Expected signature" theorem, we construct a distance matrix wherein the entries correspond to the Euclidean Distance between the signature of each time series and all others. We then transform this distance matrix into a similarity matrix using a strictly monotone decreasing function, specifically $F(x) = \frac{1}{a+x}$ with $a = 1$. In order to ascertain the most significant time series, we conduct Principal Component Analysis (PCA) [18] on the similarity matrix. This analysis enables us to identify the most significant eigenvalue and its associated time series. We generate the graph associated to the selected time series using visibility graph algorithm.
The concept underlying the visibility graph entails representing time series values as vertical bins, wherein each bin connects to others if an unobstructed line of sight exists from one bin's top to another's. Therefore, given a discrete time series $S = \{s_0, s_1, \ldots, s_T\}$ with its associated set of discrete timestamps $\mathcal{T} = \{t_0, t_1, \ldots, t_T\}$, the visibility condition can be defined using the following equation: $s_k < s_j + (s_j - s_i)\frac{t_j - t_k}{t_j - t_i}$. Meaning that any two data points (s_i, t_i) and (s_j, t_j) are connected in the graph if any other data point (s_k, t_k) satisfies this condition. This approach yields a connected graph, with the number of nodes corresponding to the observations within the time series. Importantly, the visibility graph is chosen for its ability to preserve both the temporal and structural characteristics of the time series. Specifically, periodic time series result in regular

graphs, random time series yield random graphs, and fractal time series produce random graphs. Additionally, the visibility graph remains invariant under affine transformations of the time series. The type of visibility condition applied determines whether the resulting graph is undirected or directed. Computation of the visibility graph is facilitated using the Python package "Time series to visibility graphs" (ts2vg) [19]. Figure 1 illustrates the functioning of the visibility graph using a time series extracted from the training set of a randomly selected truck. The first plot displays the path trajectories of the time series, the middle plot represents the visibility condition, and the last plot depicts the associated undirected graph. For a more comprehensive understanding, including a formal definition and visual representations, refer to [3].

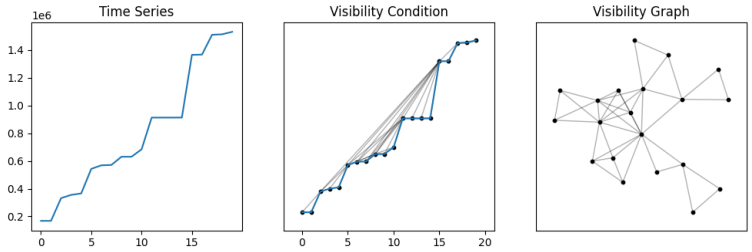

Fig. 1. The visibility graph technique is applied to feature 171 of a randomly selected truck from the training dataset.

3 Exploratory Data Analysis

Initially, we conducted an exploratory analysis of the features. This step was crucial due to the lack of information about the origin of these features, caused by the strong anonymization of the dataset. In this exploratory analysis description, we also highlight some issues mentioned in [11] that, in our opinion, should be corrected in a future version. Recall that the dataset describes 14 attributes measured at non-homogeneous timesteps for a fleet of vehicles. Six of these attributes are divided into bins related to a certain condition imposed on that attribute, and the columns represent the values of these bins. The remaining six attributes are numerical counters.

The first analysis aimed to verify that NaNs appear jointly in attributes, meaning that a NaN can appear at time t for vehicle i in attribute j if and only if all other bins for attribute j at time t for vehicle i are NaN. This is consistent with the fact that all bins of the same attribute measure the same type of data.

According to the paper, the features are cumulative, meaning the values in the columns should be increasing when restricted to individual vehicles. However, this property only holds for intervals of values without NaNs. In the paper, NaNs

are explained in section 2.1.1, where it is particularly mentioned that "the data collection counters could be reset, i.e., start again from the beginning", but in reality, the value restarts from zero only in 3.4% of cases. If the columns were increasing, or if the reason why they might not be was known, it would be possible to interpolate missing data. Instead, the lack of information makes imputation very difficult.
Another issue detected concerns the initial values of the features. If the timestep column measures "the duration in time_step that each vehicle has been utilizing Component X during its operational lifespan," it is reasonable to assume that at time 0, the features should have a value of 0. This is not the case, see for example vehicle 5.
The analysis also revealed that, only in train_operational, in 7 cells the non-decreasing property described above is not verified. This was interpreted as a transcription error or artificial noise added during the compilation and anonymization of the dataset, and it was corrected by linear interpolation. Note that in val_op and in test_op, this does not occur.
Multiple statistics on validation_operational_data and test_operational_data were evaluated and compared. These statistics showed that from all statistically evaluable perspectives, the data distribution is similar, which is expected from a dataset initially constructed for a challenge. This suggests that in future versions of the dataset, it might be useful to artificially introduce a slight distribution shift in the test, to better simulate real-world applications.
This analysis also allowed us to identify a significant outlier in the length of the time series (number of timesteps per vehicle) in test operational data. While in the validation, the maximum length is 209, and in the test, it is 262, there is only one vehicle in test with this length, all the others being at most 209 timesteps long. This observation was used to reduce the computational complexity of the training set during preprocessing.

4 Data Preprocessing

First, a new column, class_label, was created using the information contained in train_tte. For each row, class_label contains the ground truth for the vehicle and the corresponding timestep. If in_study_repair = 0, class_label = 0, otherwise class_label is 4,3,2,1,0 if length_of_study_time_step - timestep is in $[0, 6)$, $[6, 12)$, $[12, 24)$, $[24, 48)$, $[48, +\infty)$, respectively. This highlighted another issue in the data collection description: When in_study_repair = 1, what did the vehicle do from the last timestep to length_of_study_time_step? If Component X continued to work until that moment, the described ground truth is correct, but in that case, measurements at time length_of_study_time_step, which are probably the most important, are missing. Otherwise, it would seem reasonable to calculate the ground truth with the last timestep.
Given the significant scale difference between the values of a single vehicle as the timestep increases, it was necessary to choose a normalization procedure. The values of bin i of attribute j were divided by the average sum of the values of all

bins of attribute j across the entire dataset. Given the semantics of the columns, this normalization represents a scaling change for attribute j.
Subsequently, the 8 columns taken from train_specifications were integrated, transformed into integers. Note that the statement that the specifications "can take categories in Cat0, Cat1, ..., Cat8" is a typo, as these columns take integer values up to 28.
NaNs were replaced with 0. Various other imputation methods were tried, but none provided satisfactory results, also due to the issues described above. The value 0, although rarely appearing as a true value in some measurements, was deemed distinctive enough to allow a model with good predictive capacity to use it.
Subsequently, for each vehicle with in_study_repair = 0, the final records were removed until the last timestep t satisfied length_of_study_time_step $-t \geq 48$. This is because the ground truth = 0 at a time $t \in [0, 48)$ might be wrong, and false negatives have a very high cost. 202454 out of 1122452 records were removed, but given the high proportion of class 0 compared to others, this did not pose problems in the subsequent choice of time series to use for training.
For training, a dataset balanced between class 0 and non-zero classes was chosen. Given the high cost of false negatives, it was preferred not to risk the model learning to distinguish 0 better than the other classes. In the visibility graph approach training was done by aggregating all positive classes as class 4.
To maintain a high number of training samples, data augmentation was performed on time series with class_label > 0, iteratively removing the last timestep until the class became 0. At the end of this augmentation, time series with class_label $= 0$ were randomly chosen and cut to a random length, sampled from the distribution of time series lengths in the test (which the EDA showed to be similar to the distribution of lengths in the validation). This data leakage should not be used for a model aiming to generalize in the real world, but it seemed reasonable to use it for a challenge. Graphs were augmented with the time-series signature as global feature.
For models that require time series of constant length, the option was to bring all series to the maximum length using zero-padding inserted at the initial timesteps. Semantically, this corresponds to performing some initial measurements when the vehicle is not yet operational, and seemed more reasonable than adding padding at the end. Thus, the maximum length should have been 262. However, since the EDA highlighted that this length is only reached by a single vehicle, for this single vehicle the option was to truncate the initial timesteps. This allowed for a significant reduction in one dimension of the training set.
Finally, a tensor X of shape (50511, 209, 114) containing the timestep (necessary because of the irregular sampling), all attribute bins (105), and specifications (8), and a vector y of shape (50511,) with the ground truths were created.

5 Experiments

Graph Neural Networks. We used the pytorch-geometric [16] GIN [17] graph neural network, with 114 input channels (the time-series features, time step, and truck specifications), $8 \times 114 = 912$ hidden channels with 15 layers and 2 output channels. We found that, due to the high cost involved in mispredicting a class with a lower one, it was worth using only class 0 and class 4 for training and classification. During training sequences that had a class in $\{1, 2, 3\}$ were treated as if they belonged to class 4. The undirected graphs on which the GNN operated were visibility graphs constructed using the 171_0 time series for the truck. The number of nodes was thus equal to the number of timesteps in the time series. Using a threshold of 0.6 for class 0, our model obtains a cost of 40109.0 on the validation set, where costs are given in [11]. In Figure 2, the confusion matrix on validation, and totals on validation and test in Figure 3.

CM	0	1	2	3	4
0	2292	0	0	0	2618
1	4	0	0	0	12
2	3	0	0	0	11
3	11	0	0	0	19
4	15	0	0	0	61

Fig. 2. Confusion matrix on validation. True labels on rows.

set	label	0	1	2	3	4
validation	#	2325	0	0	0	2721
test	#	2314	0	0	0	2731

Fig. 3. Number of predictions for each class on validation and test sets.

ROCKET. In the realm of time series analysis, the classification model known as ROCKET (RandOm Convolutional KErnel Transform) stands out for its remarkable efficiency and accuracy. Stemming from [1], ROCKET [5] generates a vast feature space by applying thousands of random convolutions over the input time series. We used ROCKET in its sktime implementation, with 10000 random kernels over fixed-length time series, see preprocessing in Section 4. This resulted in an embedding space with 20000 dimensions, where we tested a Ridge classifier, and a fully connected model augmented with multi-bias and global connection as in [21, 22]. However, the final performance was below the baseline, and is not reported here. This is somewhat surprising, because ROCKET is considered state of the art for multivariate time series classification. While this is an indication of the complexity of the dataset, we think this low performance could be due to how the training set was built, and deserves future investigation.

6 Conclusion

We show that the Component X dataset presents a significant challenge for PdM and time-series analysis, marking a valuable contribution to the field. We anticipate that updates to the foundational paper [11] will incorporate our feedback. Our future work aims to expand the use of signature-augmented data beyond graph-based models. Moreover, we believe that adopting a multilayered graph approach, as outlined in [20, 23], could enhance graph prediction techniques.

References

1. Guang-BinHuang, Qin-Yu Zhu, Chee-Kheong Siew: Extreme learning machine: Theory and applications. **Neurocomputing**, 70(1-3), 489-501. (2006)
2. Scarselli, F., Gori, M., et al.: The Graph Neural Network Model. **IEEE Transactions on Neural Networks**, 20(1), 61-80. (2008)
3. Lacasa, L., Luque, B., et al.: From Time Series to Complex Networks: the Visibility Graph. **Procs of the National Academy of Sciences**, 105(13), 4972-4975. (2008)
4. Wu, Z., Pan, S., et al.: A Comprehensive Survey on Graph Neural Networks. **IEEE Transactions on Neural Networks and Learning Systems**, 32(1), 4-24. (2020)
5. Dempster, A. and Petitjean, F., and Webb, G.I.: ROCKET: Exceptionally fast and accurate time series classification. In: **arXiv:1910.13051**. (2019)
6. Lyons, T.: Rough Paths, Signatures and the Modelling of functions on Streams. In: **arXiv preprint arXiv:1405.4537**. (2014)
7. Chen, K. T.: A Faithful Representation of Paths by Noncommutative Formal Power Series. In: **Transactions of the AMS**, vol. 89(2), pp. 395-407. (1958)
8. Levin, D., Lyons, T., Ni, H.: Learning from the Past, Predicting the Statistics for the Future, Learning an Evolving System. In: **arXiv:1309.0260**. (2016)
9. Gregnanin, M., De Smedt, J., et al.: Signature-Based Community Detection for Time Series. In: **Studies in Computational Intelligence**, vol 1142. (2024)
10. Gregnanin, M., De Smedt, J., et al.: Stock Price Time Series Foresting using Dynamic Graph Neural Networks and Attention Mechanism in Recurrent Neural Networks. To appear in: **MIDAS - ECML-PKDD**. (2023)
11. Kharazian, Z., Lindgren, T., et al.: SCANIA Component X Dataset: A Real-World Multivariate Time Series Dataset for Predictive Maintenance. In: **arXiv:2401.15199**. (2024)
12. Lyons, T., Ni, H.: Expected Signature of Brownian Motion up to the First Exit Time. In: **The Annals of Probability**, vol. 43(5), pp. 2729-2762. (2015)
13. Chevyrev, I., Lyons, T.: Characteristic Functions of Measures on Geometric Rough Paths. In: **The Annals of Probability**, vol. 44(6), pp. 4049–4082 (2016)
14. Levin, D., Lyons, T., Ni, H.: Learning from the Past, Predicting the Statistics for the Future, Learning an Evolving System. In: **arXiv:1309.0260**. (2013)
15. Chevyrev, I., Kormilitzin, A.: A primer on the Signature Method in Machine Learning. In: **arXiv preprint arXiv:1603.03788**. (2016)
16. Fey, M., Lenssen, J.E.: Fast Graph Representation Learning with PyTorch Geometric. In: **ICLR Workshop on Representation Learning**. (2019)
17. Xu, K., and Hu, W., Leskovec, J., Jegelka, S.: How Powerful are Graph Neural Networks?. In: **arXiv preprint arXiv:1810.00826**. (2018)
18. Tipping, M. E., and Bishop, C. M: Probabilistic Principal Component Analysis. In: **Journal of the Royal Statistical Society: Series B**, 61(3), 611-622. (1999)
19. Time Series to Visibility Graphs (ts2vg) Python Packages, https://cbergillos.com/ts2vg
20. Lacasa, L., Nicosia, V., Latora, V.: Network structure of multivariate time series. In: **Sci Rep** 5, 15508 (2015).
21. Metta, C., Fantozzi, M., et al.: Increasing biases can be more efficient than increasing weights. In: IEEE/CVF Winter Conference on Applications of Computer Vision WACV (2024)
22. Di Cecco, A., Metta, C., Fantozzi, M., Morandin, F., Parton, M.: GloNets: Globally Connected Neural Networks. To appear in LNCS, IDA proceedings (2024)
23. Freitas Silva, V., Eduarda Silva, M., et al.: MHVG2MTS: Multilayer Horizontal Visibility Graphs for Multivariate Time Series. In: **arXiv:2301.02333**. (2023)