14th International Conference on Applications of Statistics and Probability in Civil Engineering, ICASP14
Dublin, Ireland, July 9-13, 2023

Unsupervised Methods for Railway Bridge Structural Monitoring

Roberto Boccagna
Research scholar, Dept. of Engineering and Geology, Univ. "G. D'Annunzio", Pescara, Italy

Maurizio Bottini
PhD student, Dept. of Engineering and Geology, Univ. "G. D'Annunzio", Pescara, Italy

Massimo Petracca
Research Engineer, Asdea Software, Pescara, Italy

Matteo Di Giorgio
Research Engineer, Asdea Hardware, Pescara, Italy

Alessia Amelio
Researcher, Dept. of Engineering and Geology, Univ. "G. D'Annunzio", Pescara, Italy

Guido Camata
Professor, Dept. of Engineering and Geology, Univ. "G. D'Annunzio", Pescara, Italy

ABSTRACT: The purpose of this paper is to present some preliminary results obtained from the application of a dedicated Artificial Intelligence-based monitoring software to an OpenSees numerical model of a railway bridge to assess its health conditions in near real-time. The proposed approach is based on the construction of an unsupervised Machine Learning algorithm in a full data-driven scenario, with the aim of establishing a reliable method for anomaly detection even in the absence of a numerical model. A reference pattern is obtained by collecting data at high sampling frequency on several fixed nodes as trains of various masses cross the undamaged bridge at different velocities; after pre-processing, the data are fed into various types of autoencoders which are trained to produce outputs as close as possible to the inputs. It is then shown that the algorithm actually flags the data produced when damage scenarios are activated in the OpenSees model as coming from a damaged structure. Our solution can be also adopted as a binary classifier once a threshold for reconstruction errors has been fixed.

1. INTRODUCTION

Artificial Intelligence (AI) and especially Deep Learning (DL) algorithms are widely employed in the Civil and Structural Engineering context since large amounts of data can now be transmitted and processed in a relatively short time [Tibaduiza Burgos et al. (2020)]. Systems for early anomaly detection draw from a broad ensemble of Machine Learning (ML) techniques, the final purpose being the ability to automatically recognize possible damage that may affect a given structure so that appropriate measures can be taken in time.

Although most data analysis methods are well established, the scientific community has yet to agree on the sequence of processes to be performed, starting from the acquisition of raw data via sensors to get reliable predictions. This is especially a pivotal issue in Structural Health Monitoring (SHM), although general consensus exists regarding AI, with specific reference to the algorithms worth adopt-

ing in unsupervised or supervised cases. In the SHM framework, we rely on unsupervised algorithms whenever the training data come from structures in healthy conditions, as the dedicated program has no way to distinguish between undamaged and damaged scenarios. In such cases, using the data available, AI tries to infer the underlying process that produces the observed output, hence learning to identify data that is statistically incompatible with that generation process, as the paradigm of SHM is that any structural change is reflected in the mechanism that produces the observed data. The dedicated literature recommends autoencorder neural networks as the type of algorithms best suited for accomplishing this task. Conversely, supervised algorithms are adopted whenever the training set also contains data corresponding to damage scenarios that may be generated by running dynamic numerical simulations using the Digital Twin (DT) of the physical structure under examination. In this case, the literature advises several multiclass DL algorithms for optimal performance [Farrar and Worden (2013)].

In the present work, we have restricted our focus to the unsupervised case, using the accelerometric time series we obtained by running railway bridge dynamics for a Finite Element Method (FEM) created using Asdea's Scientific ToolKit for OpenSees (STKO) advanced GUI for OpenSees [Petracca et al. (2017)] for both the training and for the test phase. This was done with the intent to establish a standard procedure for managing strictly data-driven contexts, from data collection to output emission, with the idea of providing a robust solution for application in real monitoring cases.

The paper is organized as follows. Section 2 presents the bridge model and provides information on the simulation set-up. Section 3 explains how raw data are pre-processed and how the features from the extracted raw data are fed as inputs to the neural network. Section 4 is dedicated to the presentation of the results, and in particular, we show how the algorithm correctly classifies undamaged and damaged data series. Section 5 examines the conclusions drawn and suggests further developments.

2. THE MODEL

2.1. STKO modelization

The model used for testing our baseline anomaly detection solution was created using the STKO interface for OpenSees, and reproduces a steel truss railway bridge with riveted connections consisting of 3 spans (Figure 1). The structure is approximately 93 m long, 5 m wide, and has two piers that are 4.8 m high. Each lateral span measures $\sim$ 28 m, while the central one is $\sim$ 35 m long. Further information about the model is provided below:

- the braces were modeled using T-profile truss elements
- piers were modeled using 4-node shell elements (ASDShellQ4)
- the ballast was modeled using springs
- tracks were modeled using IPE beam elements
- both top and bottom chords were modeled using double-T beam elements
- verticals were modeled using IPE300 beam elements
- diagonals were modeled using double-C beam elements

Regarding boundary conditions, the piers were fixed to the base; one of the two abutments blocks linear displacements along the x, y and z directions and rotations about the x axis, while the other abutment blocks linear displacements along the y and z axis. The spans are attached to the piers by means of rigidLinks OpenSees elements.

The monitoring system of the bridge consists of a network of 12 triaxial accelerometric sensors, which sample at a rate of 1 kHz, located at highly representative nodes as depicted in Figure 1.

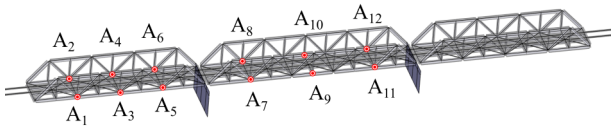

Figure 1: Schematic representation of the bridge and sensor positions.

Damage scenarios can be introduced by reducing the elastic moduli of 32 different structural elements by modifying them as a percentage from 0 to

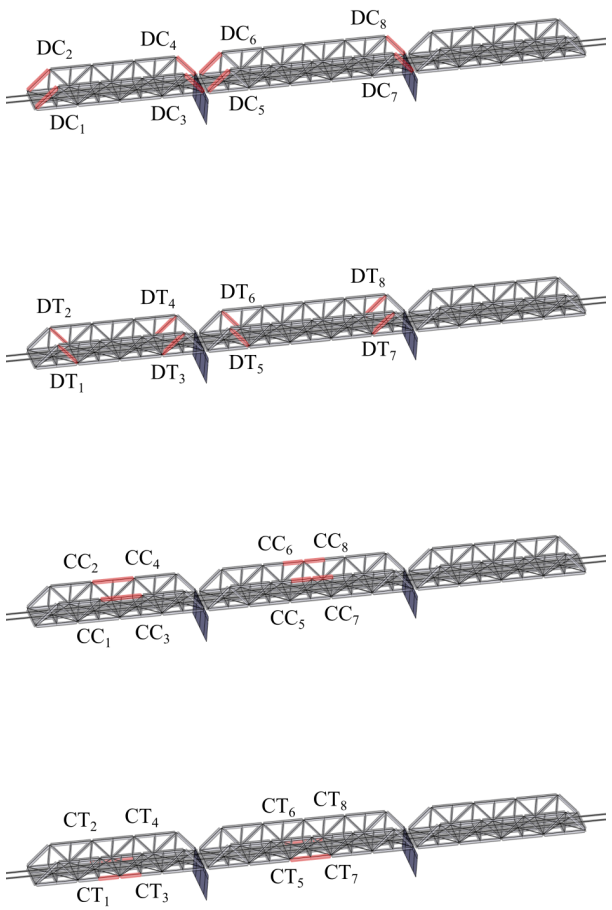

Figure 2: Locations of the 32 elements whose elastic modulus the user can reduce. Legend: DC (diagonal in compression), DT (diagonal in tension), CC (chord in compression), CT (chord in tension).

100, giving the possibility to determine which elements to damage. Figure 2 shows the elements for which the user can reduce the elastic modulus, consisting of 8 diagonals in compression, 8 diagonals in tension, 8 chords in compression and 8 chords in tension.

2.2. *Numerical Simulations*

Numerical simulations were performed as follows. A single run refers to the passage of one train across the railway bridge which generates a single instance in terms of the AI algorithm. Partially following Parisi et al. (2022)'s strategy, trains were modeled using their mass and velocity, represented as random variables extracted from log-normal distributions with parameters respectively $\mu_{\text{mass}} = 62$ ton, $\sigma_{\text{mass}} = 5$ ton and $\mu_{\text{velocity}} = 8.33$ m/s, $\sigma_{\text{velocity}} = 1$ m/s, while the length of the trains is constant and fixed to 40 m. A single run lasts for $T = 50$ s, regardless of the exact moment the caboose of the train crosses the last bridge element, to obtain consistent time windows for processing. The output of the STKO program consists of accelerometric time series considering the components along x, y and z axis for each of the 12 control points, for a total of 36 time series for each run.

3. THE ALGORITHM

3.1. *Feature Extraction*

Univariate analysis was applied to each time series to aggregate the information embedded in the entire run through a minimal set of parameters. This is the first step in eliminating redundant data. A collection of temporal, spectral and statistical features was extracted using the Python TSFEL library [Barandas et al. (2020)] as suggested in [Buckley et al. (2022)][1]. In particular, a set of 163 features was selected for each of the 36 accelerometric series obtained for each run, considering the whole set of default features the library computes after removing a large number of Fourier coefficients to discard frequencies higher than 30 Hz. The features obtained were then arranged in a matrix A of dimensions $N_{\text{runs}} \times (N_{\text{features}} \times N_{\text{sensors}})$, where $N_{\text{features}} = 163$ and $N_{\text{sensors}} = 36$ in this case.

3.2. *Feature Reduction*

The second step for database reduction consists of the simple application of a linear PCA algorithm at fixed retained variance.

3.2.1. *Training Phase*

Dataset standardization is first performed on A in such a way that

$$A_{ij} \mapsto \frac{A_{ij} - \mu_j^A}{\sigma_j^A} \quad \text{for any } i, j \tag{1}$$

[1] See `https://tsfel.readthedocs.io/en/latest/` for a complete list of computed features.

where μ_j and σ_j are, respectively, the mean and the standard deviation computed along the j-th column of A. After covariance matrix $C = \frac{1}{N_{\text{runs}}-1}A^\top A$ is computed, the spectral decomposition of $C = O\Lambda O^\top$ is performed, being $\Lambda = \text{diag}\left(\lambda_1, \ldots, \lambda_{N_{\text{features}} \times N_{\text{sensors}}}\right)$ the matrix containing the eigenvalues of C in non-increasing order[2]. A is then projected along the first n components, which retain a given amount of the total variance (99% in our case):

$$X = AP_n\,, \quad P_n = O\left[:, :n\right]\,, \tag{2}$$

where (we used Python notation) P_n is the $N_{\text{runs}} \times n$ matrix containing the first n columns of O.

3.2.2. Test Phase

Once the algorithm is trained, new data are standardized as in (1) using the same μ_j^As and σ_j^As computed over the matrix A. Then, the dataset is projected along the n eigendirections, multiplying by matrix P_n as in (2).

3.3. The Neural Network

The rows $\{\mathbf{x}_i\}_{i=1}^{N_{\text{runs}}}$ of matrix X represent single input instances for the neural network. The first algorithm tested was a vanilla autoencoder (AE) with input dimension n and just one hidden intermediate layer, which represents the latent space and has dimension $d = \lfloor n \rfloor + 1$. This choice was motivated by the purpose of testing one of the most simple neural network architecture before performing an optimal hyperparameters choice through dedicated tools. The graph is fully connected, as depicted in Figure 3, and a hyperbolic tangent is used as an activation function.

In the training phase, the normalized sum of Euclidean distances between the inputs and outputs was adopted as a loss function. Let $f_{\theta,\phi}\left(\mathbf{x}_i\right)$ be the reconstructed version of input $\mathbf{x}_i$, where θ and ϕ indicate the collection of parameters of the encoding and decoding portions respectively. For the reconstruction error, we take the following quantity:

$$e_{\theta,\phi}\left(\mathbf{x}_i\right) = \left\|\mathbf{x}_i - f_{\theta,\phi}\left(\mathbf{x}_i\right)\right\|_2\,. \tag{3}$$

[2] C is symmetric positive semi-definite, its eigenvalues are non-negative, and its eigenvectors form an orthogonal basis for $\mathbb{R}^{N_{\text{features}} \times N_{\text{sensors}}}$.

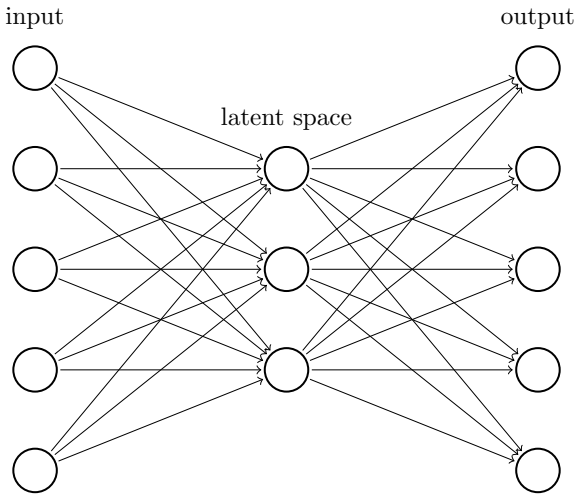

Figure 3: Schematic representation of the vanilla autoencoder used.

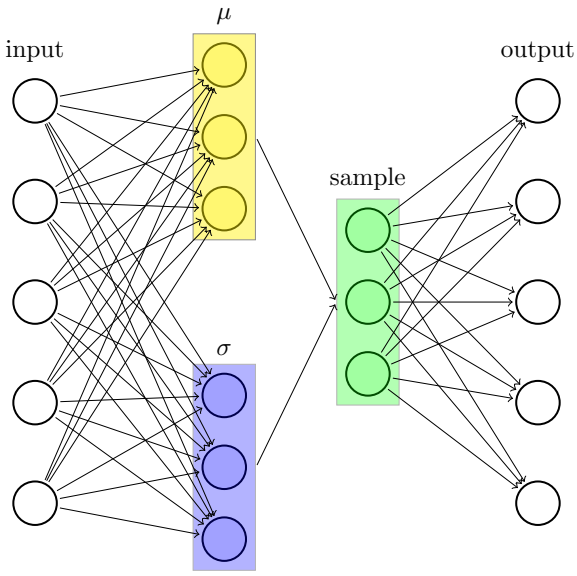

Figure 4: Schematic representation of the variational autoencoder used.

The loss function is obtained by averaging the whole training dataset:

$$L^{\text{AE}}\left(X\right) = \frac{1}{N_{\text{runs}}} \sum_{i=1}^{N_{\text{runs}}} e_{\theta,\phi}\left(\mathbf{x}_i\right)\,, \tag{4}$$

so that biases and weights converge to the values that best allow the network to get outputs as similar as possible to the inputs belonging to the training set in the ℓ^2 norm.

The second neural network adopted was a variational autoencoder (VAE), whose architecture is

described in Figure 4. Unlike the simple vanilla autoencoder, the latent representation is not deterministic, as the latent vector is sampled from a multivariate, Gaussian distribution with mean vector μ and diagonal covariance matrix σI. Let $p_\theta(\mathbf{z})$ equal the prior probability of obtaining the latent vector, $\mathbf{z}$, $p_\theta(\mathbf{z} \mid \mathbf{x}_i)$ equal the posterior distribution of $\mathbf{z}$ given $\mathbf{x}_i$, and $p_\theta(\mathbf{x}_i \mid \mathbf{z})$ be the conditional probability of $\mathbf{x}_i$ given $\mathbf{z}$, where θ again represents the collection of encoder parameters. The true posterior distribution is generally intractable and is then approximated by a distribution of $q_\phi(\mathbf{z} \mid \mathbf{x}_i)$ (ϕ the collection of parameters for the decoder), which is chosen to be Gaussian, and represents the probability of observing the output $\mathbf{x}_i$ given the latent variable $\mathbf{z}$. The *evidence lower bound* (ELBO) is typically adopted as a loss function for VAEs, being a mixture of cross-entropy between the original and reconstructed dataset and Kullback-Leibler divergence that measures the functional distance between the true prior and the approximated posterior (see Kingma and Welling (2013) for derivation):

$$L^{\text{VAE}}_{\theta,\phi}(X) = \frac{1}{N_{\text{runs}}} \sum_{i=1}^{N_{\text{runs}}} \ell_{\theta,\phi}(\mathbf{x}_i), \tag{5}$$

$$\ell_{\theta,\phi}(\mathbf{x}_i) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i \mid \mathbf{z})] - \mathbb{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}_i) \| p_\theta(\mathbf{z})\right). \tag{6}$$

The first term at the right-hand side of (6) can be estimated through a reparametrization trick, while Kullback-Leibler divergence assumes a simple expression after forcing $p_\theta(\mathbf{z})$ to be a standard Gaussian on $\mathbb{R}^d$. For details, we again recommend consulting [Kingma and Welling (2013)].

For the sake of notational simplicity, hereafter, we will use $e_i \equiv e_{\theta,\phi}(\mathbf{x}_i)$.

3.4. Statistics of the Reconstruction Error

From a probabilistic point of view, the collection of reconstruction errors $\{e_i\}_{i=1}^{N_{\text{runs}}}$ represents a set of independent identically distributed random variables on $\mathbb{R}^+$, as single train passages are mutually independent. The resulting statistics are a generalized chi-square distribution because components of $\mathbf{x}_i$ and its reconstructed counterpart are, in principle, correlated. Various methods can be applied for deducing probability distribution $p(e_i)$ from data, thus establishing a threshold for distinguishing undamaged from damaged data in the inference phase. Once threshold $\alpha \in (0,1)$ is established, a given instance can be considered anomalous when its reconstruction error e is such that $p(e) < \alpha$. We used kernel density estimation (KDE) method for inferring the pdf of the underlying process. In our application, the bandwidth h of the kernel, which is the main parameter of the method, was estimated via Silverman's rule of thumb $h = 0.9 \min\left(\sigma_e, \frac{\text{IQR}}{1.34}\right) N_{\text{runs}}^{-\frac{1}{5}}$, where σ_e is the standard deviation of reconstruction errors, and IQR represents the interquartile range. The threshold for acceptability α was fixed to 0.005, a value that finds its validity a posteriori. Note that, in this initial phase of the construction of our framework, we expect a structure in a healthy state to produce false alarms with probability α. To further perfect the framework, a method for alarm validations is thus required and will be introduced in the future.

4. RESULTS

The training of the two algorithms was performed using $N_{\text{runs}} = 500$ train passages, consequently inferring the reconstruction error pdf and obtaining values for $e_{\min}$ and $e_{\max}$ such that $p(e < e_{\min}) < \alpha$ and $p(e > e_{\max}) > \alpha$. Figure 5 shows the shape of the pdfs deduced from the training data for AE and VAE, alongside the corresponding frequency histogram.

We initially focused on the predictive capability of the two algorithms as anomaly detectors by introducing damage on node DC_1. Specifically, FEM dynamics were run after the elastic modulus of the mentioned beam element was reduced by a factor of 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, and 0.10, considering 10 runs for each of these percentages of damage. In addition, we ran 25 further simulations for the undamaged bridge in order to check if the trained algorithms would classify the corresponding inputs as non-anomalous. Figure 6 shows the collection of reconstruction errors for the training set (from 0 to 499, blue colored) and for the test set corresponding to the 25 undamaged data sets (500 to 524, green colored if regular, red colored if anomalous), using thresholds (the dashed, gray

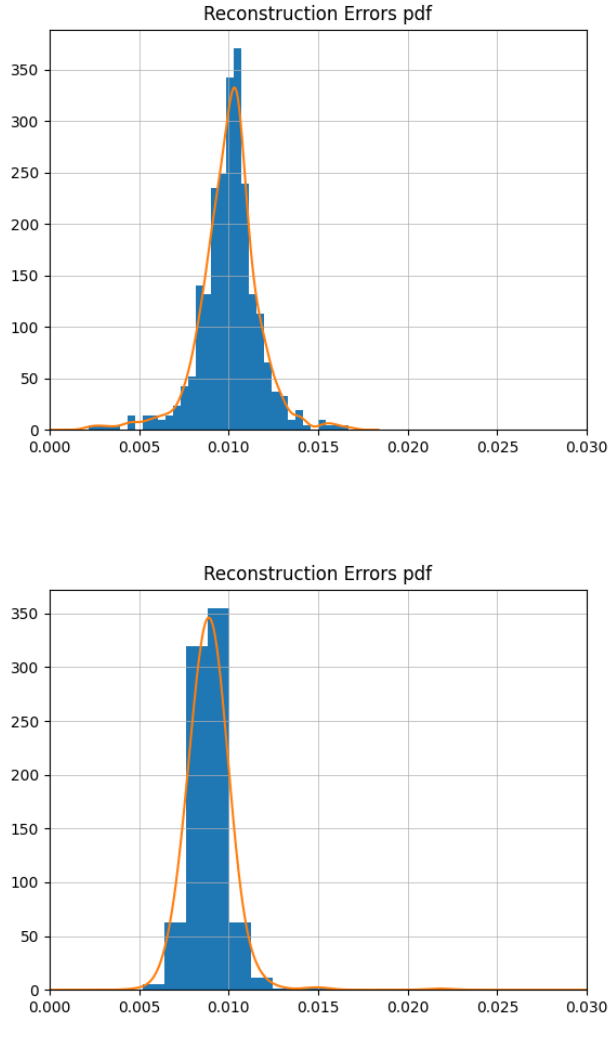

Figure 5: The pdf (the orange lines) inferred from training data in the case of AE (upper) and VAE (bottom). Histograms (in blue) were obtained using 20 bins.

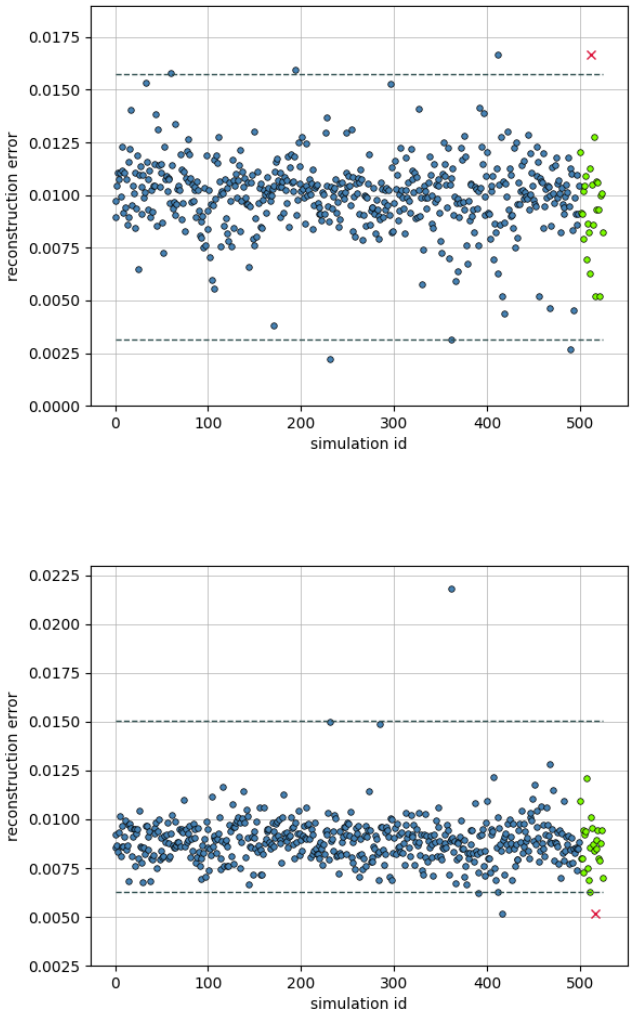

Figure 6: The reconstruction error for training and non-damaged test sets for AE (top) and VAE (bottom).

lines) corresponding to the values of $e_{\min}$ and $e_{\max}$. For the AE, the mean reconstruction error of the training set is $\mu^{\text{AE}}_{\text{training}} \simeq 1.00 \cdot 10^{-2}$, compared with a value of $\mu^{\text{AE, undamaged}}_{\text{test}} \simeq 9.53 \cdot 10^{-3}$ for the set of the 10 undamaged tests. For the VAE, we obtained $\mu^{\text{VAE}}_{\text{training}} \simeq 8.90 \cdot 10^{-3}$ and $\mu^{\text{VAE, undamaged}}_{\text{test}} \simeq 8.55 \cdot 10^{-3}$. Both the average reconstruction error values calculated for the test sets are statistically compatible (regarding the statistics deduced from training data) with the means of the training sets (significantly within 1σ).

Figure 7 shows the collection of reconstruction errors for the training set and for the afore-mentioned damage configurations (enumerated from 500 to 559, red colored, in increasing damage order), adopting the log-scale for the y-axis since reconstruction errors spread on various scales. Notice that both algorithms recognize all the "damaged" instances as anomalous, although the reconstruction error appears to be non-monotone in the elastic modulus reduction. It is also worth noting that VAE is, in this case, more damage-sensitive compared to the AE-based solution since the errors corresponding to the test set cover a larger interval.

This is also a consequence of VAE's pdf appearing more peaked around its maximum. The computational costs are comparable for the two neural network autoencoders employed.

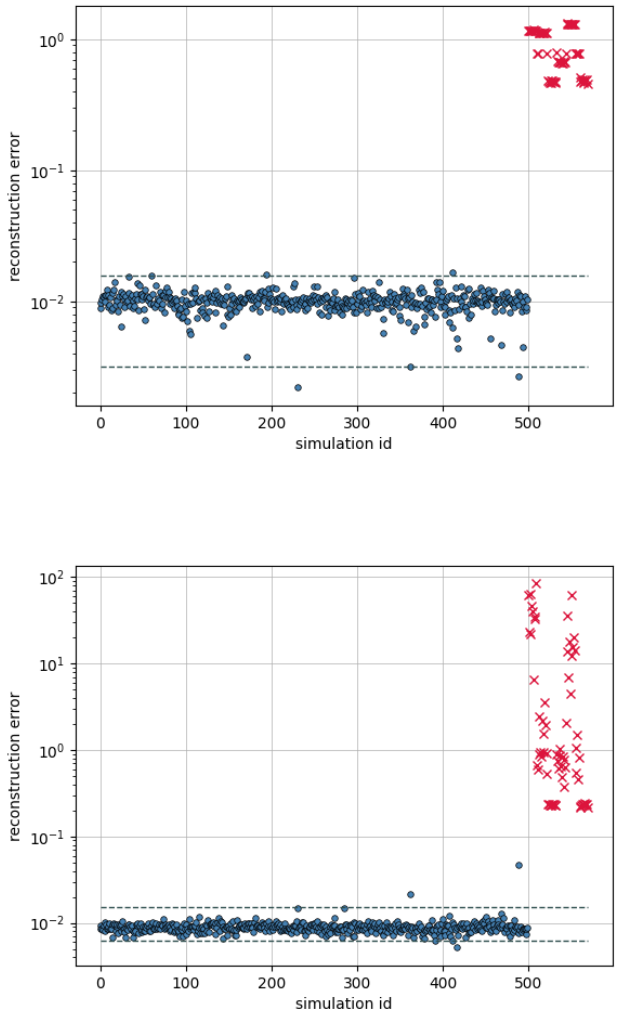

Figure 7: The reconstruction error for training and damaged test sets as indicated in the text for AE (top) and VAE (bottom).

We performed a second test running numerical dynamics simulations for each of the 32 damage locations as in Figure 2. In order to have strong data consistency, we fixed the mass and the velocity of the train crossing the bridge to the mean values of 62 tons and 8.33 m/s and reduced the elastic moduli of beam elements by a factor of 0.02. Figure 8 again illustrates reconstruction errors for the training sets and for the 32 different damage scenarios. The evidence supports the conclusion that our solutions correctly classify all the instances given as inputs as anomalous.

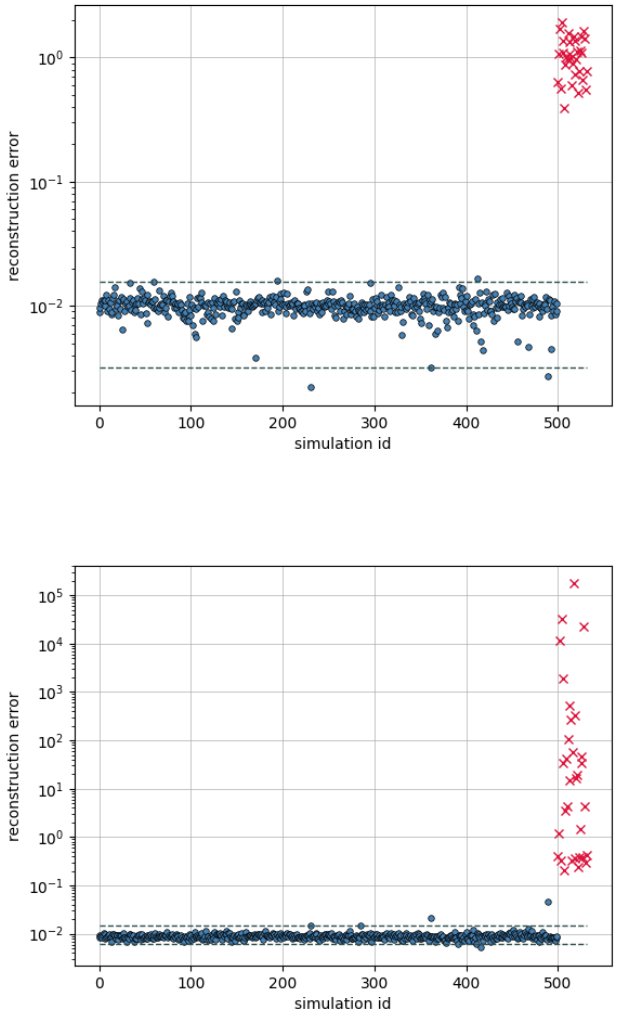

Figure 8: The reconstruction error for training and the second damaged test sets as indicated in the text for AE (top) and VAE (bottom).

5. CONCLUSIONS

This work represents a simple yet reliable framework for SHM in the full data-driven case. The results obtained so far are promising as our detector succeeded in recognizing healthy states and classifying the various configurations of damage types/severity as anomalous. Despite this, we were

unable to define a connection between elastic modulus reduction and the output obtained. In the future, more effort will be made towards selecting features and exploring the space of the neural network's hyperparameters to enable our solution to estimate the actual damage level from the value of the reconstruction error obtained. After that, we can move towards considering the supervised case and trying to identify possible damage locations from test data. The introduction of a digital twin of the monitored structure would also improve the overall accuracy of the solution.

6. REFERENCES

Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., Liu, H., Schultz, T., and Gamboa, H. (2020). "Tsfel: Time series feature extraction library." *SoftwareX*, 11.

Buckley, T., Bidisha, G., and Pakrashi, V. (2022). "A feature extraction & selection benchmark for structural health monitoring." *Structural Health Monitoring*.

Farrar, C. R. and Worden, K. (2013). "K. structural health monitoring: A machine learning perspective." *Wiley: Oxford, UK*.

Kingma, D. P. and Welling, M. (2013). "Auto encoding variational bayes." *arXiv preprint arXiv:1312.6114*.

Parisi, F., Mangini, A. M., Fanti, M. P., and Adam, J. M. (2022). "Automated location of steel truss bridge damage using machine learning and raw strain sensor data." *Automation in Construction*, 138.

Petracca, M., Candeloro, F., and Camata, G. (2017). "Stko user manual." *ASDEA Software Technology, Pescara, Italy*.

Tibaduiza Burgos, D. A., Gomez Vargas, R. C., Pedraza, C., Agis, D., and Pozo, F. (2020). "Damage identification in structural health monitoring: A brief review from its implementation to the use of data-driven applications." *Sensors*, 20(3).