# STRUCTURAL HEALTH MONITORING OF THE HOCHZOLLERN-BRIDGE USING COMPUTATIONAL INTELLIGENCE: A CASE STUDY

**Abhishek Nain[1] (452987), Marcelo Martinez[2] (390809), Praveen Mohanram[3] (external Student)**

**[1]Management and Engineering in Computer Aided Mechanical Engineering MSc.**
**[2]General Mechanical Engineering MSc.**
**[3]Fraunhofer Institute for Production Technology**

**ABSTRACT:** Structural health monitoring (SHM) involves analysing a given structure and detecting damage if present. This report explores the application of artificial intelligence for SHM, using the Hochzollern-bridge in Cologne as a case study. The objective is to develop models able to detect imperfections in FEM simulations of the bridge. Three different models are presented and compared: one with classical machine learning using random forest classification and two neural networks. While the random forest achieved an accuracy over 91%, it was outperformed by the deep learning models, which got over 99% and also a faster prediction time. In addition, for the neural networks, the results on the test set show adequate generalization capabilities. The report also discusses different approaches to handling input data with unequal lengths, as the simulations have a different amount of nodes, and the importance of selecting appropriate input features from the FEM simulations.

**KEYWORDS:** Structural Health Monitoring, Machine Learning, Neural networks, Feature Extraction.

## 1. INTRODUCTION:

Structural Health Monitoring (SHM) is essential for securing the safety and reliability of buildings by preventing potential failures. Traditionally, sophisticated models such as Finite Element Analysis (FEA) and manual inspections of critical areas have been employed. However, there may be better approaches, given the dynamic nature of structural health. In response, advancements in technology, including sensors and machine learning, offer real-time monitoring solutions, predicting failures, issuing alarms for safety, and enabling prompt corrective measures for structural integrity. Sensors such as vibration sensors and strain gauges are used to collect valuable information such as deformation on the axis and strain on the node points of the structure. These data are then stored and analyzed using various AI methods to predict the structure's condition accurately.

This report focuses on training an artificial intelligence (AI) model designed for classifying structural conditions using a provided dataset. The dataset encompasses information on two distinct types of structures of the Hochzollern Bridge in Cologne, Germany: perfect and imperfect. Each structure is identified by multiple numbered nodes, with raw data capturing deformation in the X, Y, and Z directions, along with shear stress in the XY, YZ, and ZX planes. The objective is to train the AI model and assess its performance using a validation dataset. Throughout the report, we explore both Machine Learning and Deep Learning models. Following an initial performance analysis, we select the most suitable model and fine-tune it to achieve optimal results. The whole application is developed and tested using python as the programming language and the ML libraries Tensorflor, sklearn and matplotlib for visualization.

**2. METHODS:** To better understand the case, Figure 1 illustrates the bridge in the FEM simulation software and compares the deformation in the perfect structure with one with missing nodes.

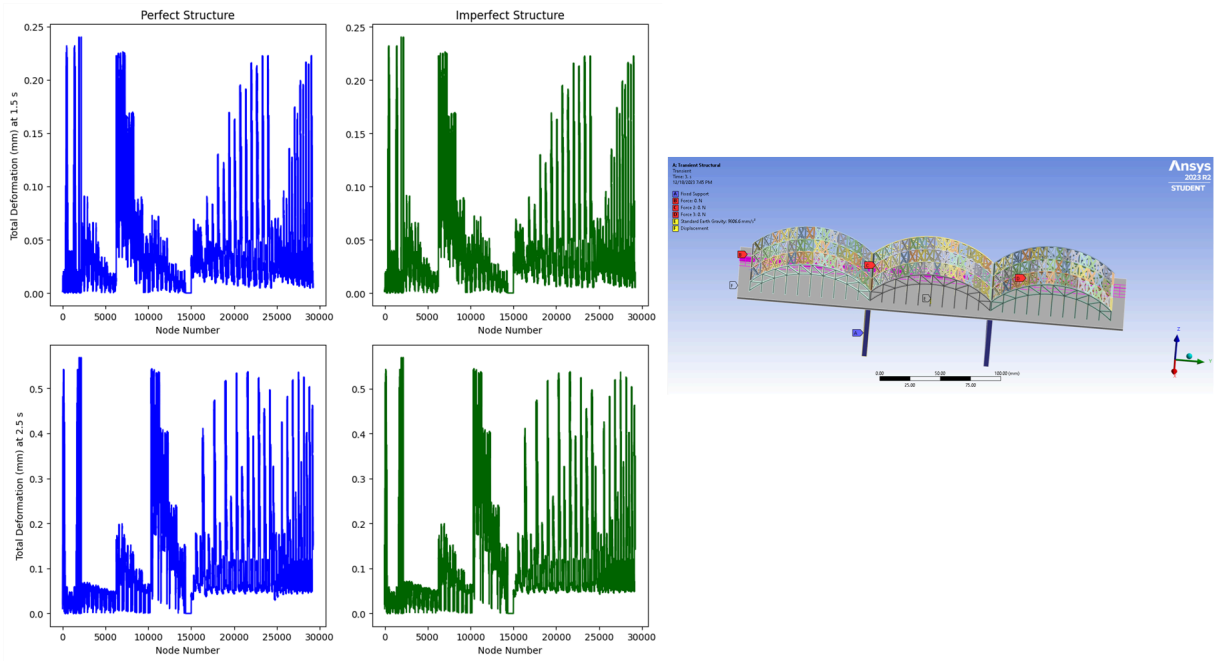


**Figure 1:**
**Left: Comparison between the total deformation for the perfect structure and one with imperfections (left). Two time points are plotted.**
**Right: 3D model of the bridge and representation of the elements**

The Figure highlights the complexity of the dataset. There are over 29000 nodes, the elements vary in length and also in their spatial orientation. In addition, the load is moving, which can be seen by the shift of the peak in deformation on the left. Also, the dataset provided not only includes the total deformation but also the deformation in each direction and the shear stress in all planes. Selecting the relevant data for the models and preprocessing is therefore necessary. We investigated three different models, one through classical machine learning and two variations using deep learning. We looked at their performance and then picked the one that worked best. Before proceeding with training, the dataset is labelled to differentiate the perfect and imperfect dataset.

### 2.1 First approach: Classical Machine Learning

**2.1.1 Feature Extraction:** Feature extraction is a crucial step in preparing data for AI model training. It involves transforming raw and often complex data into a more manageable, relevant, and informative representation, commonly known as features. These extracted features enhance the model performance compared to a raw data set. Various features can be extracted from the dataset, but the following are selected for the model training.

Mathematical stats: This involves applying mathematics statistics such as mean, variance and standard deviation to categorize the dataset. These provide insights such as variability between the data.
Fast Fourier Transforms (FFT): It is a mathematical algorithm used for transforming time or spatial domain signals into the frequency domain. FFT helps analyze the frequency components of data, revealing patterns or trends
Principal Component Analysis: It is a dimensionality reduction technique. It transforms the original variables of a dataset into a new set of uncorrelated variables called principal components. In feature extraction for AI models, PCA helps reduce the dimensionality of the data while retaining its essential information, enhancing efficiency and mitigating the risk of overfitting.

**2.1.2 Random Forest Classifier:**
Upon examination of the dataset, it is clear that using a time series data analysis model, such as classification or regression, is enough. But, using a simple model could lead to overfitting or underfitting issues. We use a robust Random Forest classifier model to overcome this challenge. In the Random Forest classifier [1], a cluster of decision trees is used to address classification and regression tasks. Instead of depending on a single decision tree, which might be overly sensitive to the training data, the Random Forest combines predictions from numerous decision trees to enhance accuracy and robustness. The operation of the Random Forest involves first random data sampling with bootstrapping. The sampled data is then employed to construct individual decision trees based on specific features from the dataset. These multiple decision trees, forming an ensemble or forest, collectively contribute to the voting process during the prediction phase. The votes from each decision tree are utilized and weighted to determine the final prediction value. This approach is more robust to underfitting and overfitting within the dataset.

The model undergoes training utilizing the combined features obtained from the previous step. The dataset is separated into training and testing subsets, with the model being fed the training data. Subsequently, the trained model is validated using the test dataset.
For the input data of Total deformation, the accuracy achieved was 91% and a F1 score of 0.91.

**2.2 Second approach: Deep Learning**

Due to the complexity of the dataset, we also decided to develop a neural network model for the task. Our goal here is to define a classification task, where each node gets a value between 0 and 1, which defines its possibility of it being missing. Two models were trained.

**2.2.1 Long Short Term Memory (LSTM) Model:** The simulations are in a time series, and the train moves throughout the structure. Therefore, the position of the force caused by the weight of the train changes relative to the nodes in each time step. LSTMs are specially suited for sequential data (Smagulova et al. 2019), so we started with this model. We used as an input the shear stress in the YZ plane as a time series for each node. The stress was normalized using scipy's MinMaxScaler. The main challenge here is that each simulation has a different amount of nodes, but the dimensions of the input of the neural network must be the same. We detected the positions of the missing nodes of the imperfect structures and padded them with an arbitrary number. Then we added a masking layer before the LSTM layer to our model. The masking layer avoids using the padded nodes in training. Finally we add a dense layer with sigmoid activation, with an output size equal to the amount of nodes of a perfect structure and a value between 0 and 1, as mentioned above. The labels for the training are taken from the simulation data.

**2.2.2 Feedforward model:** After developing the LSTM model there were two questions to be responded: Could the classification be done with a simpler model? Is there an alternative strategy for padding that could result in a better generalization? To answer these questions, we designed a third model using a feedforward neural network. Given that the bridge is either defective or perfect from the beginning of the simulation, here we take a different approach when preparing the data. Instead of looking at the time series, we take each time step of the simulation as an individual input. The advantage compared to the LSTM is that this way the batch size increases significantly. We also have to deal with the unequal input lengths, but we modified the padding strategy. Instead of directly filling the values of the missing nodes with an arbitrary value in the place where they should be, we take the values of all the nodes in their original order. The values are scaled with the MinMaxScaler and then we pad the

shorter vectors towards the end to match the length of the largest input, which corresponds to the simulation with no defects. This approach should improve the generalization capabilities of the model. Finally, instead of the shear stress, we use the total deformation of each node. The output is exactly the same as in the LSTM model.
For the network structure, we start with a masking layer to hide the padded values. Then we add a dropout layer with a frequency of 0.2 to avoid overfitting (Data Base Camp 2023). Following that we include 4 dense layers with Relu activation and we finalize with a dense layer with sigmoid activation for the classification. A comparison of both models is illustrated in Figure 2.

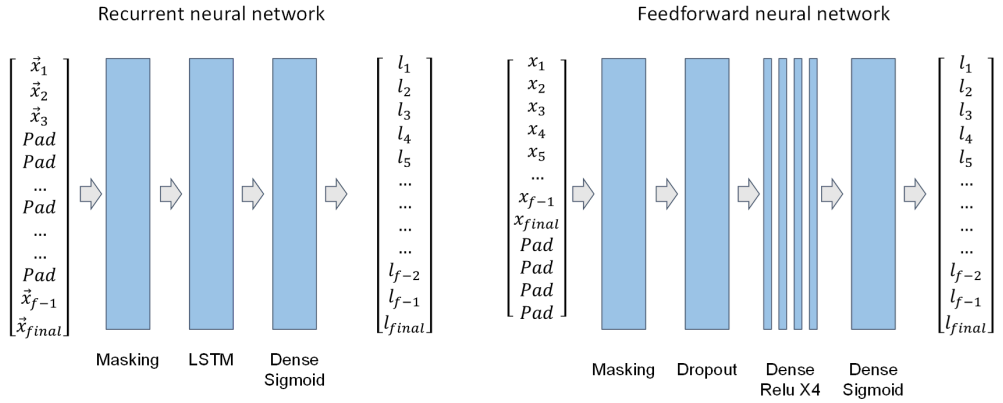


**Figure 2: Comparison between the architecture of the recurrent (left) and feed forward (right) neural network architectures. The input of the RNN is the shear stress YZ as a time series, the padding is performed to fill the missing information of the missing nodes. The input of the feedforward NN is the total deformation of one timepoint. The padding is performed after all the values of the nodes.**

For the LSTM, the data for training and validation is split in 80% and 20% respectively. For the feedforward, because of the larger dataset, the split is 70% and 30 %. Before the split, the data was shuffled to make sure that the models were trained using both the perfect and imperfect structures. One simulation was left out for posterior testing. The goal here is to have a case completely unknown for the models to see their generalization capabilities.

**3. RESULTS:** Comparing the models, the three performed well with the dataset provided. The machine learning model (Random Forest Classifier) got an accuracy of 91%, but the training time was longer. Additionally, expanding the features for classification proved to be too complex for the ML model to train and classify. The neural network models, which automatically extract features as they train, can handle the dataset and features without creating overhead on the computational resources, making them more efficient for training and also achieving better accuracy. Therefore, in the following table, we will focus on these models.

In Table 1 the results of both models are compared :

| | | Feedforward model | LSTM model |
|---|---|---|---|
| Input | | Total deformation for one time point for each node | Shear stress XY as a time series for each node |
| Scaling | | MinMaxScaler | |
| Training configuration | Loss | Binary crossentropy | |
| | Epochs | 200 | 84 (early stopping) |
| | Learning rate | 0.001 | 0.01 |
| | Batch size | 48 | 10 |
| Training | Accuracy | 100% | 99.9% |
| | F1 score | 100% | 96.5% |
| Testing | Accuracy | 100% | 100% |
| | F1 score | 100% | 100% |

**Table 1: Comparison of the training configuration and results of the LSTM and Feedforward Models**

The accuracy and f1 score are calculated by binary classification, which means there is a decision boundary. If the output of the network is below 0.5 we assume there is a node. If it is above, the network predicted a missing node. In Figure 3 the loss over the epochs is illustrated for both models. It can be seen that in neither case the validation loss increases towards the end. Therefore, we can assume that both models did not overfit.

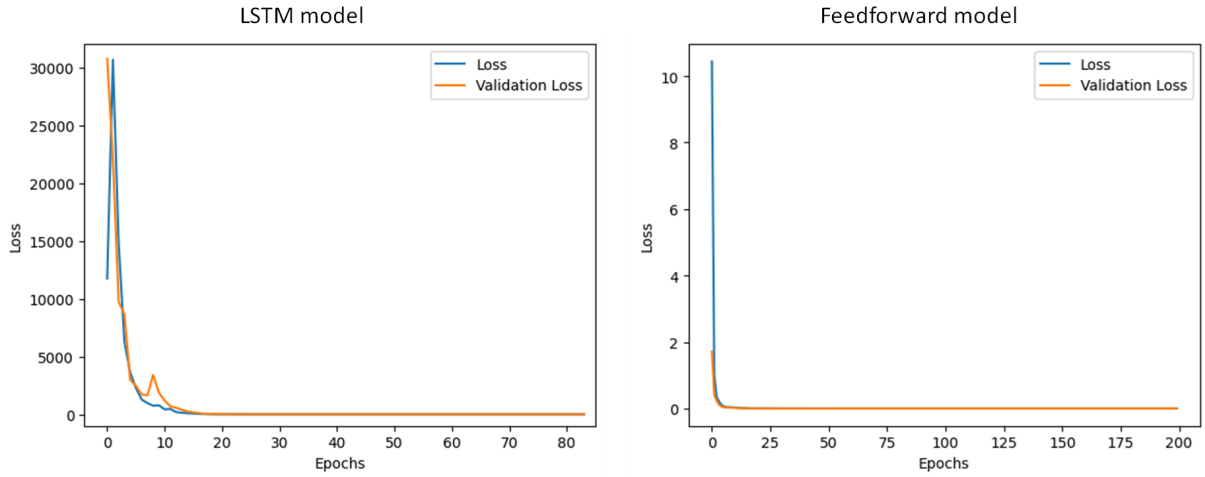


**Figure 3: Comparison of the loss over epochs for both deep learning models**

**4. DISCUSSION:** The DL approaches showed better results than the random forest classifier. In addition, in DL, using the model for predictions is much faster than classifying using random forest. The computational effort is only large when training the network, which is done only once. When it comes to comparing both DL approaches, the feedforward model had an improved performance compared to the LSTM. The feedforward model got a perfect f1 score for training and validation, indicating no false positives or negatives, whereas the LSTM did not. In addition, the feedforward model only requires the measurements of one time step, instead of the whole time series. This means that less data needs to be collected for the feed-forward model to predict the state of the bridge. Regarding the padding strategies, both performed well on the test set. However, it is possible that padding directly where the missing nodes should be, as it is implemented for the LSTM model, may result in the network not being able to learn appropriately. This effect is caused because in each input the masked values are exactly on the same position as the corresponding labels with

imperfections. We therefore find the second strategy, padding towards the end, more suitable.

Finally, when choosing what input is best for the neural networks, it is important to take a look at how FEM simulations work. The structure is divided into a number of elements.They can be expressed in the matrix form of equation F = KX (Similar to Hooke's spring law). Here K is known as the stiffness matrix, X is the vector of displacements, and F is the vector of forces. The stiffness matrix is calculated based upon the boundary conditions and design of the particular elements. The above is then used to calculate the nodal displacements. Stress and strain are then post-processed from the displacement solution. Their accuracy can vary depending on the type of method used for their calculation. Therefore it makes more sense to use nodal displacement as input for the neural network, like it was done in the feedforward model.

**5. CONCLUSION:** The study presented here extends the use of AI in SHM by classifying structural conditions through data derived from FEA simulations. The classical machine learning approach shows good accuracy, but the deep learning models, particularly the feedforward neural network, outperform in terms of efficiency and precision. The dataset utilized, though derived from simulations, might introduce some discrepancies compared to real-world sensor data, thereby suggesting for future validation efforts with actual sensor inputs. Challenges such as noise and uncertainties in real-world data pose opportunities for refining and adapting the models.

**6. REFERENCES:**

Breiman L.(2001), Random Forest. Machine Learning, 45(1), 32-5.
Data Base Camp (2023) What is the dropout layer? https://databasecamp.de/en/ml/dropout-layer-en#:~:text=The%20dropout%20layer%20is%20a,the%20network%20architecture%20at%20all.
Smagulova, K., & Pappachen James, A. (2019) A survey on LSTM memristive neural network architectures and applications DOI:10.1140/epjst/e2019-900046-x
Tao, J. & Bang Yun, C (2019). A review on deep learning-based structural health monitoring of civil infrastructures. Smart Structures and Systems, Vol. 24, No. 5 (2019) 567-586 DOI: https://doi.org/10.12989/sss.2019.24.5.567
Worden, K. & Manson, G (2006). The application of machine learning to structural health monitoring. Phil. Trans. R. Soc. A (2007) 365, 515–537 doi:10.1098/rsta.2006.1938

## 7. Contribution of team members in various work areas:

- Data analysis - Abhishek
- Data conditioning - Abhishek, Marcelo
- Pre-processing - Abhishek, Praveen
- Evaluate various AI algorithms with dataset - Marcelo (DL), Praveen (ML)
- Optimizing and fine-tuning - Marcelo (DL), Praveen (ML)
- Validate and estimate parameters - Marcelo (DL), Praveen (ML)
- Presentation - Abhishek, Marcelo
- Report - Abhishek, Marcelo, Praveen

**Abhishek Nain**

**Marcelo Martinez**


**Praveen Mohanram**