ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Chair of Structural Mechanics
and Monitoring

Semester Project

Damage localization in a masonry arch bridge based on acceleration data

Application of machine learning methods using modal parameters and time-series data for classification and localization of damage

January 2025

Supervised by:
Dr. Charikleia Stoura
Dr. Xudong Jian

Author:
Simon Scandella

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Damage localization in a masonry arch bridge based on acceleration data

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Simon	Scandella

Student supervisor(s)

Charikleia	Stoura
Xudong	Jian

Supervising lecturer

Eleni	Chatzi

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Zürich, 31.01.25
Place and date

Scandella
Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

This project explores machine learning techniques for structural health monitoring (SHM) to detect and localize damage in masonry arch bridges. Modal parameters, including natural frequencies and mode shapes, were extracted from accelerometer data using methods such as Stochastic Subspace Identification (SSI),Frequency Domain Decomposition (FDD) and Eigensystem Realization Algorithm (ERA). Additionally, the raw accelerometer data was also utilized directly. Of the ML methods tested the Random Forest (RF) model using the FDD dataset with zone-based labeling achieved strong performance in damage detection (92.11% accuracy and 90.25% recall) and promising results in damage localization, though low recall in some zones highlighted challenges with class imbalance. The Convolutional Neural Network attention model (CNN-attention) using ERA-derived modal parameters also showed decent results for damage detection. However, his model struggled with damage localization using an image based labeling scheme which made it necessary to split the data chronologically. Thus it failed to extrapolate on high-level damage scenarios in the test set.
Additionally, a 1D-2D CNN model trained on time-series data using an image-based labeling scheme was tested. However, similar to the ERA-dataset pipeline, the need for chronological data splits combined with the scarcity of high-damage scenarios in early tests caused the model to struggle with generalization. Incorporating noise-augmented high-damage scenarios into the training set improved performance, underscoring the model's struggles with extrapolating to high-damage scenarios when trained solely on low-damage scenarios. This has to be noted as a form of data leakage which introduce a bias in terms of the generalization ability of the model. Overall, the neural network models exhibited signs of overfitting, highlighting the limited amount of data, particularly the lack of granularity in damage scenarios between the healthy state and failure. In contrast, the RF models demonstrated robustness by achieving strong results despite the small dataset.
These findings underscore both the potential and limitations of applying machine learning methods for SHM.

Symbols

Acronyms and Abbreviations

CNN	Convolutional Neural Network
ERA	Eigensystem Realization Algorithm
FDD	Frequency Domain Decomposition
MAC	Modal Assurance Criterion
MLP	Multi-Layer Perceptron
RF	Random Forest
SE	Squeeze-and-Excitation
SHM	Structural Health Monitoring
SSI	Stochastic Subspace Identification
SSI-COV	Covariance Driven Stochastic Subspace Identification

Chapter 1

Introduction

Nothing lasts forever, and masonry arch bridges are no exception. Over time, these structures experience wear and tear due to natural forces and human activity. To ensure their continued safety and usability, regular maintenance is essential. This project focuses on applying machine learning techniques to perform structural health monitoring (SHM) by detecting and localizing damage in a masonry arch bridge.

Using data from a study conducted at Leeds University by Liu et. al. [1], which documented accumulated damage to a full-scale masonry arch bridge under laboratory conditions, this project applied machine learning models in a supervised manner to detect and localize structural damage. SHM is particularly useful for structures that are difficult to inspect manually and can minimize the impact of human error. Swift damage detection and precise localization can reduce long-term maintenance costs and improve safety.

Two approaches were explored: First, modal parameters (e.g., natural frequencies and mode shapes) where extracted from the accelerometer time-series data using methods such as covariance driven Stochastic Subspace Identification (COV-SSI), Frequency Domain Decomposition (FDD), and Eigensystem Realization Algorithm (ERA). Second, the accelerometer data was directly used as input for prediction. These approaches aim to advance the integration of SHM with modern machine learning techniques.

Chapter 2

Methods

2.1 Data Acquisition

The data for this project was acquired by the experiment conducted in Liu et. al. [1]. In the following chapter an overview of the experimental setup will be given. For a more detailed explanation of the experiment reading the study's paper is recommended.

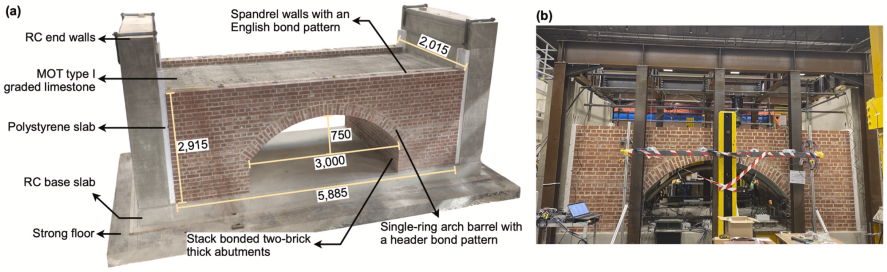


Figure 2.1: (a) The masonry arch bridge with dimensions in mm; (b) testing site. Picture from [1] page 2.

Four accelerometers (A1, A2, A3 and A4) recording acceleration over time in three spatial directions where placed on the bridge. Data from the accelerometers where recorded for a 60 second time frame (sampling frequency of 200 Hz) during which the bridge was excited ten times with a hammer at a specific location on the bridge. Data was recorded separately for the 5 different excitation points L1, L2, L3, L4 and L5. First, in Test 1 the healthy state was recorded. The bridge was than put under loads causing damage to the bridge and accelerometer data was recorded again. This was repeated until the bridge failed. The loads applied to the bridge in between data recording increased repeatedly. Finally, we have a data set with five CSV files for each damage state from the healthy state (Test 1) to right before the bridge failed (Test 25). Issues with applying the loads in Test 23 and Test 24 led to missing data for these two damage states in the provided dataset. Ultimately, the dataset is composed of 115 CSV data files with a timestamp column and 12 accelerometer data columns (four sensors with each three channels for three spatial directions). Each CSV files is specific for one of the five excitation locations (L1 to L5). The following figure 2.2 shows the placement of the accelerometers and the location of the excitations.

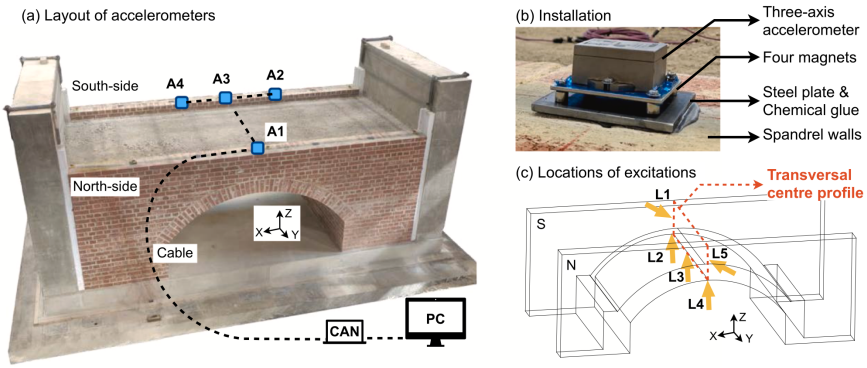


Figure 2.2: (a) Accelerometer layout; (b) Shows one accelerometer; (c) Locations of the excitations. Picture from [1] page 4.

2.2 Labels

In this project a supervised approach was chosen.
In Liu et. al. [1] the damage to the masonry arch bridge was recorded by drawings depicting the damage for each test. An example from Liu et. al. can be seen in figure 2.3.

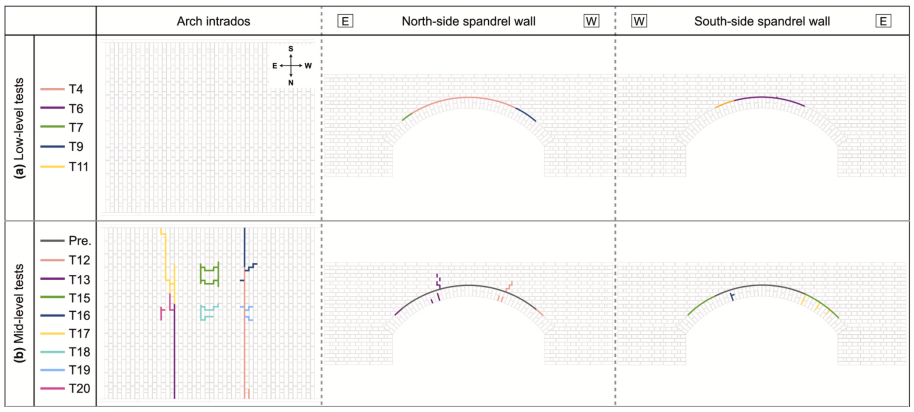


Figure 2.3: A portion of the damage recorded after each test. Picture from [1] page 7.

A zoning approach was selected as the initial labeling method. The bridge was divided into ten zones, with each zone assigned a binary label for each test. A value of one indicates the presence of new damage, while a value of zero indicates no damage. This approach results in a multi-class binary label matrix, where each row corresponds to a specific test and each column represents a zone. A more granular version with 20 zones was also tested. An example of the zoning approach with 10 zones is illustrated in figure 2.4 and with 20 zones in figure 2.5.
The zoning approach serves as an effective initial labeling method due to its simplicity and ease of integration into a machine learning pipeline. Its formulation as a multi-class binary classification problem allows for straightforward application of standard supervised learning algorithms.
This method has certain limitations. One notable drawback is that the label for a zone must be set to 1 regardless of the extent of damage within the zone. For

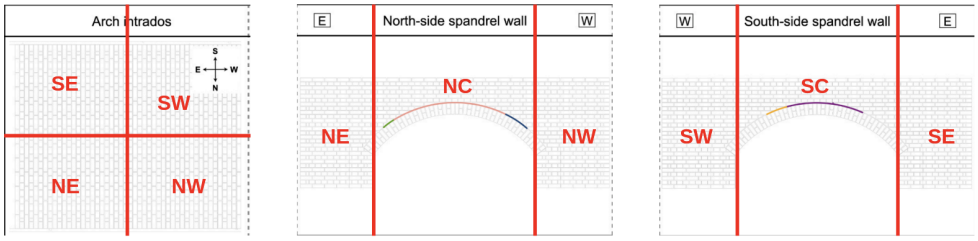


Figure 2.4: The bridge divided into 10 zones. Background picture adapted from [1] page 7

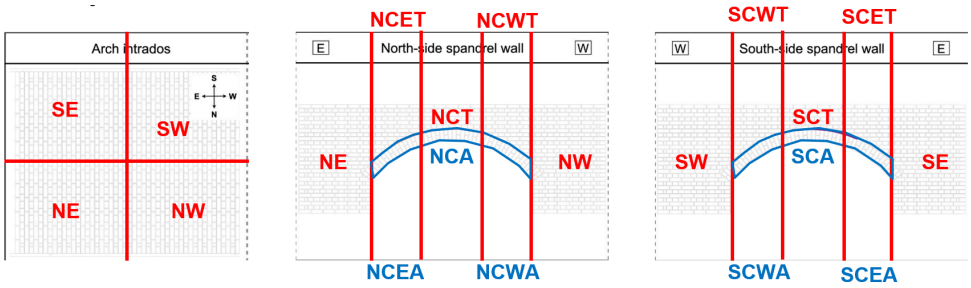
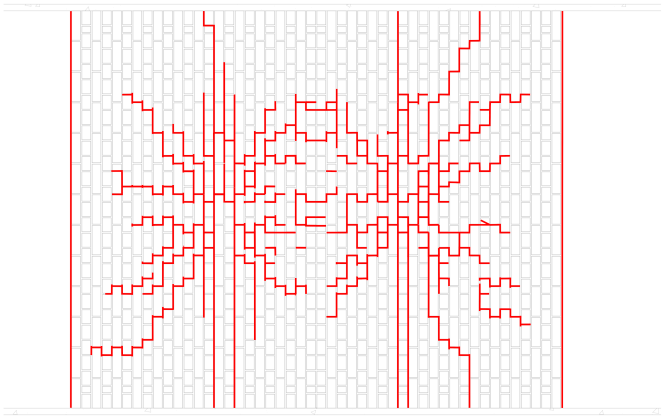
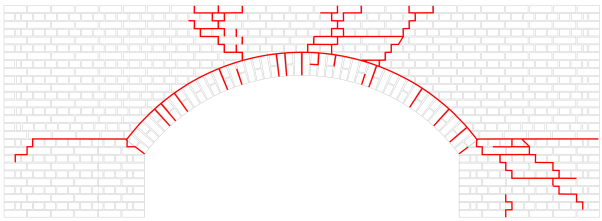


Figure 2.5: The bridge divided into 20 zones. Background picture adapted from [1] page 7

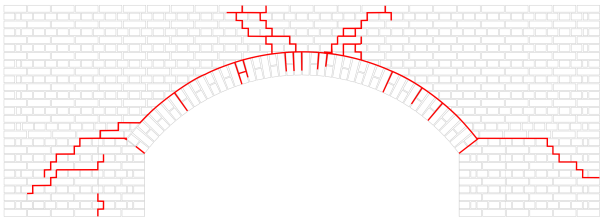
instance, if a crack spans two zones with 95% of its length in one zone and only 5% in the adjacent zone, both zones will be assigned a label of 1. This lack of granularity can reduce the precision of damage localization.
In Roy et. al. [2], the authors state that "the difference in damaged and undamaged mode shapes is barely observable. However, the change in this mode shape and the change in derivatives of this mode shape are clearly showing the presence and location of damage" (Journal of Sound and Vibration 332 p. 5590, Fig. 5(a)). While derivatives of mode shapes provide a clearer indication of damage presence and location, detecting small-scale damage within a specific zone is expected to be challenging due to the subtle changes in mode shape.
To address these limitations, a second labeling approach was implemented. This approach is image-based, where the accumulated damage up to each test is visually represented for three perspectives: The arch intrados, the north spandrel wall, and the south spandrel wall. The damage is marked as red lines on the images using the open-source software GIMP. For each test, the three perspective images were loaded, resized to 256 by 256 pixels, and combined into a single image of size 256 by 768 pixels. To preserve the original aspect ratios, padding was applied to the images.
From this combined image, a binary mask was generated. Pixels corresponding to red (indicating damage) were assigned a value of 1, while all other colour pixels were assigned a value of 0 (indicating no damage). To make the labels less sparse, Gaussian smoothing was applied to the binary mask. This method enables the application of machine learning algorithms to map input data to binary masks, allowing for a direct comparison between the ground truth masks and the predicted damage maps. In the following figure 2.6 the image labels for test 25, the last test, are shown as an example.



(a) Image label of Test 25: Arch Intrados.



(b) Image label of Test 25: North Spandrel Wall.



(c) Image label of Test 25: South Spandrel Wall.

Figure 2.6: Image labels of Test 25 for 3 perspectives.

2.3 Modal Parameter Extraction

One approach chosen in this project was to extract modal parameters such as natural frequencies and mode shapes from the raw accelerometer data and use that as the input to a machine learning model. The reason for doing so is that, in theory, when a structure is damaged it becomes less stiff resulting in a natural frequency drop for the same structural mode. This is described in detail in "Detection of structural damage through changes in frequency: A review" by O. S. Salawu [3]. In addition, with damage to the structure, how the mode shapes and its derivatives change should contain information on the location of the damage. This is described in detail in "Fundamental mode shape and its derivatives in structural damage localization" by Roy et. al. [2]. In conclusion, it should be possible to train a machine learning model to classify and localize damage by feeding it the natural frequencies and mode shapes. In this projects three methods have been conducted to extract natural frequencies and mode shapes from the accelerometer data. The methods used are Stochastic Subspace Identification (SSI), Frequency Domain Decomposition (FDD) and Eigensystem Realization Algorithm (ERA). In the following section these three methods are described.

2.3.1 SSI - Stochastic Subspace Identification

In this project, a readily available MATLAB script for performing SSI-COV (covariance-driven stochastic subspace identification) on time-series data was downloaded from MathWorks and used as a baseline. The script, developed by Prof. Cheynet, served as an excellent starting point for further development and analysis [4]. The script was further adapted for this project's use case.
Now, an overview of the SSI-COV method is presented. This explanation is based on the paper "Understanding Stochastic Subspace Identification" by Brincker et al. [5] and Liu et. al.'s explanation of the SSI method [1] as well as the algorithm by Prof. E. Cheynet [4]. For a more detailed explanation, readers are encouraged to refer to the original papers and the algorithm.

We start with the system's stochastic response $\mathbf{y(t)}$ recorded at discrete time steps. This would be in this projects case the accelerometer data.

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_N(t) \end{bmatrix} \tag{2.1}$$

Where N is the number of channels.
The unbiased covariance matrix at a time lag k is computed as:

$$\mathbf{R}_k = \frac{1}{N-k}\mathbf{Y}_{1:N-k}\mathbf{Y}_{k:N}^T \tag{2.2}$$

where N is the total number of time steps, $\mathbf{Y}_{1:N-k}$ is the data matrix where we removed the last k data points and $\mathbf{Y}_{k:N}$ the data matrix where we removed the first k data points.
Using the covariance matrices, a Block Hankel matrix is constructed as:

$$\mathbf{H} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{R}_2 & \cdots & \mathbf{R}_j \\ \mathbf{R}_2 & \mathbf{R}_3 & \cdots & \mathbf{R}_{j+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_i & \mathbf{R}_{i+1} & \cdots & \mathbf{R}_{i+j-1} \end{bmatrix} \tag{2.3}$$

The "past" responses ($\mathbf{H}_p$) correspond to the upper half portion, and the "future" responses ($\mathbf{H}_f$) correspond to the lower half portion.
Next the future responses are projected onto the past responses:

$$\mathbf{O} = (\mathbf{H}_f \mathbf{H}_p^T)(\mathbf{H}_p \mathbf{H}_p^T)^{-1} \mathbf{H}_p \tag{2.4}$$

Here, $\mathbf{O}$ is the projection matrix that encapsulates the system's dynamic behavior. SVD is applied to the projection matrix $\mathbf{O}$:

$$\mathbf{O} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2.5}$$

where:

- $\mathbf{U}$: Left singular vectors.
- $\mathbf{\Sigma}$: Singular values.
- $\mathbf{V}$: Right singular vectors.

With this we can compute the observability matrix $\mathbf{\Gamma}$:

$$\mathbf{\Gamma} = \mathbf{U}\mathbf{\Sigma}^{1/2} \tag{2.6}$$

And the system matrix $\mathbf{A}$:

$$\mathbf{A} = \mathbf{\Gamma}^{\dagger}\mathbf{\Gamma}' \tag{2.7}$$

where $\mathbf{\Gamma}^{\dagger}$ is the pseudo-inverse of $\mathbf{\Gamma}$, and $\mathbf{\Gamma}'$ is the shifted observability matrix. The eigenvalues and eigenvectors of the system matrix $\mathbf{A}$ are computed using eigenvalue decomposition:

$$\mathbf{A}\mathbf{\Psi} = \mathbf{\Psi}\mathbf{\Lambda} \tag{2.8}$$

where:

- $\mathbf{\Lambda}$: Diagonal matrix of eigenvalues.
- $\mathbf{\Psi}$: Modal eigenvectors.

From the eigenvalues λ_i, modal parameters are extracted:

- Natural frequencies:
$$f_i = \frac{\mathrm{Im}(\lambda_i)}{2\pi\Delta t} \tag{2.9}$$
- Damping ratios:
$$\zeta_i = -\frac{\mathrm{Re}(\lambda_i)}{|\lambda_i|} \tag{2.10}$$
- Mode shapes:
$$\mathbf{\Phi} = \mathbf{C}\mathbf{\Psi} \tag{2.11}$$
where $\mathbf{C}$ is the output matrix.

2.3.2 FDD - Frequency Domain Decomposition

This section presents an overview of the FDD method. This explanation is based on the paper "Modal Identification of Output-Only Systems Using Frequency Domain Decomposition" by Brincker et al. [6]. For a more detailed explanation, readers are encouraged to refer to the original paper. For this projects application of FDD, a MATLAB script to perform FDD on the data was provided by my supervisor Dr. Xudong Jian which adapted a script by Mohammad Farshchin [7]. The initial script was automated and adapted.
We start with the system's stochastic response $\mathbf{y}(t)$ recorded at discrete time steps. In this project's case, this corresponds to the accelerometer data:

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_N(t) \end{bmatrix} \tag{2.12}$$

where $y_i(t)$ is the response of the i-th measurement channel at time t, and N is the number of sensors.
Next, the Power Spectral Density (PSD) matrix of the measured responses is computed using the Fourier transform:

$$\mathbf{G}_{yy}(j\omega) = \mathbf{H}(j\omega)\mathbf{G}_{xx}(j\omega)\mathbf{H}^H(j\omega) \tag{2.13}$$

where:

- $\mathbf{G}_{yy}(j\omega)$: PSD matrix of the output responses.
- $\mathbf{H}(j\omega)$: Frequency response function (FRF) matrix.
- $\mathbf{G}_{xx}(j\omega)$: PSD matrix of the input forces (assumed to be white noise).
- ^{H}: Hermitian (conjugate transpose).

At each frequency ω_i, the PSD matrix $\mathbf{G}_{yy}(j\omega_i)$ is decomposed using Singular Value Decomposition (SVD):

$$\mathbf{G}_{yy}(j\omega_i) = \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{U}_i^H \tag{2.14}$$

where:

- $\mathbf{U}_i$: Singular vectors (mode shapes).
- $\mathbf{\Sigma}_i = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N)$: Diagonal matrix of singular values (σ_i).

The first singular vector $\mathbf{u}_1$ at the peak frequency provides an estimate of the mode shape:

$$\mathbf{Œ} = \mathbf{u}_1 \tag{2.15}$$

The corresponding peak frequencies in the singular value spectrum represent the natural frequencies (f_n).
In summary, the FDD method decomposes the PSD matrix into singular values and singular vectors. Peaks in the singular value spectrum correspond to natural frequencies, and the associated singular vectors estimate the mode shapes.

2.3.3 ERA - Eigensystem Realization Algorithm

In the following section, the third and last method used in this project to extract modal parameters, the Eigensystem Realization Algorithm (ERA), is described. For this project, a MATLAB script written by Prof. Dr. Eleni Chatzi was used as a baseline. The script was automated and adapted for usage. This section is based on a tutorial by ETH [8] explaining the ERA method.
The ERA is a system identification method based on time-domain data, primarily used to extract modal parameters such as natural frequencies, damping ratios, and mode shapes. The key steps of the ERA are outlined below.
We start with the system's stochastic response $\mathbf{y}(t)$ measured at discrete time steps. This response is the output of the system when excited by an impulse or random input. The discrete-time system can be written as:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \tag{2.16}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \tag{2.17}$$

where:

- $\mathbf{x}(k)$: State vector at time step k,
- $\mathbf{A}$: State transition matrix,
- $\mathbf{B}$: Input matrix,
- $\mathbf{u}(k)$: Input vector,
- $\mathbf{y}(k)$: Output vector (measured response),
- $\mathbf{C}$: Output matrix,
- $\mathbf{D}$: Feedthrough matrix.

Next, the discrete impulse response of the system, $\mathbf{h}(k)$, is assembled into a block Hankel matrix:

$$\mathbf{H}_0 = \begin{bmatrix} \mathbf{h}(1) & \mathbf{h}(2) & \cdots & \mathbf{h}(j) \\ \mathbf{h}(2) & \mathbf{h}(3) & \cdots & \mathbf{h}(j+1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{h}(i) & \mathbf{h}(i+1) & \cdots & \mathbf{h}(i+j-1) \end{bmatrix} \tag{2.18}$$

This matrix, $\mathbf{H}_0$, contains the impulse response coefficients of the system and is used to capture its dynamic behavior.
Singular Value Decomposition (SVD) is then applied to the block Hankel matrix to separate the system dynamics from noise:

$$\mathbf{H}_0 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2.19}$$

where:

- $\mathbf{U}$: Left singular vectors,
- $\mathbf{\Sigma}$: Singular values,
- $\mathbf{V}$: Right singular vectors.

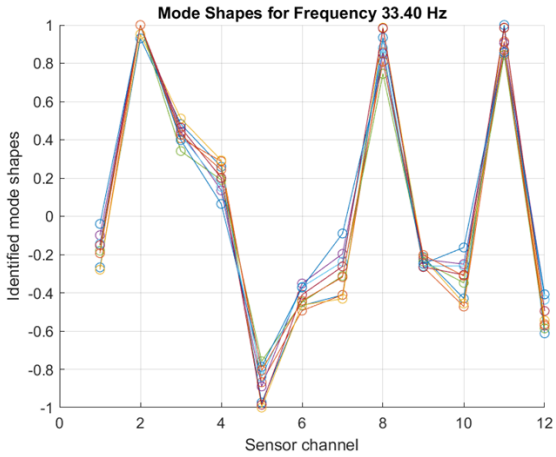
Since the subsequent steps for estimating the modal parameters follow the same procedure as in the SSI method, they will not be reiterated here.
In summary, the ERA method identifies the system's dynamic properties by analyzing the discrete impulse response and performing singular value decomposition and eigenvalue decomposition.

2.4 Structural Mode Clustering

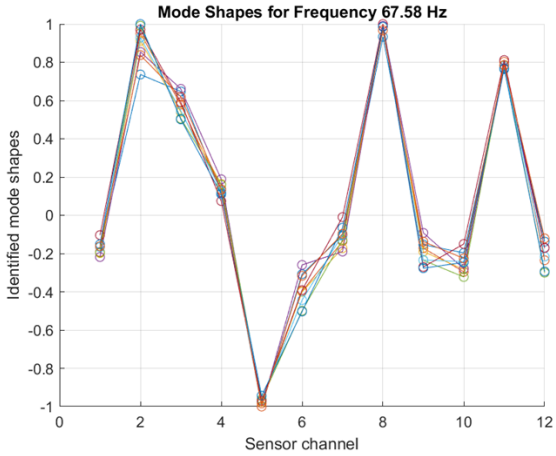
The goal of using modal parameters was to identify structural modes, defined as natural frequency–mode shape pairs that persist throughout the dataset. These modes are inherent to the structure and characterize its response. According to [3] and [2], changes in modal parameters corresponding to the same structural modes under varying damage scenarios provide critical information for damage detection and localization.

However, finding consistent structural modes across the datasets extracted using SSI, FDD, and ERA proved to be challenging. One reason for this difficulty was that the modal parameter datasets contained samples at similar frequencies, but the corresponding mode shapes were different. This made it difficult to isolate structural modes and identify which samples should be considered noise.

Additionally, some unusual data patterns were observed. For example, as shown in Figure 2.7, two mode shapes were extracted for the same test using FDD. These mode shapes appear very similar, yet they were extracted at completely different frequencies. Although this outcome is possible, it could also be an artifact of the FDD algorithm.



(a) Mode extracted using FDD at 33.4 Hz.



(b) Mode extracted using FDD at 67.58 Hz.

Figure 2.7: Modes extracted using FDD at different frequencies.

Since the extraction of structural modal parameters proved to be more challenging than expected, two clustering approaches were implemented to differentiate structural modes from noise.
The first method aimed to cluster structural modes by identifying representative mode shapes in the healthy state. These were either selected manually by plotting the mode shapes and identifying recurring patterns (for the initial pipeline using the SSI-dataset) or chosen using a script that calculates pairwise MAC scores (for the pipeline using the FDD-dataset). Representative mode shapes were selected based on the MAC values and the number of occurrences in the healthy state data.
In the second method, the ERA modal parameter dataset was utilized. Modal parameters for each test and excitation were initially clustered based on MAC values and relative frequency changes, resulting in clusters of modal parameters with similar frequencies and mode shapes for each test. These clusters were generated within the ERA MATLAB script.
Next, in a separate MATLAB script, these mode clusters were further grouped across the entire dataset to assign Cluster IDs. Clustering was performed using MAC values as the primary criterion and an absolute frequency tolerance as the secondary criterion. To account for the expected frequency drop due to damage, a frequency tolerance of 1 Hz above and 15 Hz below the values observed in the first test where the mode appeared was applied (because we expect a frequency drop with damage to the structure). Structural modes were defined as modes appearing in at least two unique tests, while modes present in only one unique test were considered noise and assigned a Cluster ID of 0. Structural mode clusters were assigned Cluster IDs greater than 0. This information was subsequently included as features in the feature vector.
As a result, with a MAC threshold of 0.98, 81.48% of the data was assigned a Cluster ID greater than 0 and was therefore considered to represent structural modes. The remaining 18.52% was assigned to the noise cluster with a Cluster ID of 0.
However, despite the clustering algorithm's results, finding consistent modes across most of the tests remained difficult. Many clusters contained samples from only two tests, while others abruptly stopped appearing in later tests or started appearing only in later tests. This underscores the challenges faced in generating a reliable modal parameter dataset.
An example of a structural mode found is shown in Figure 2.8.

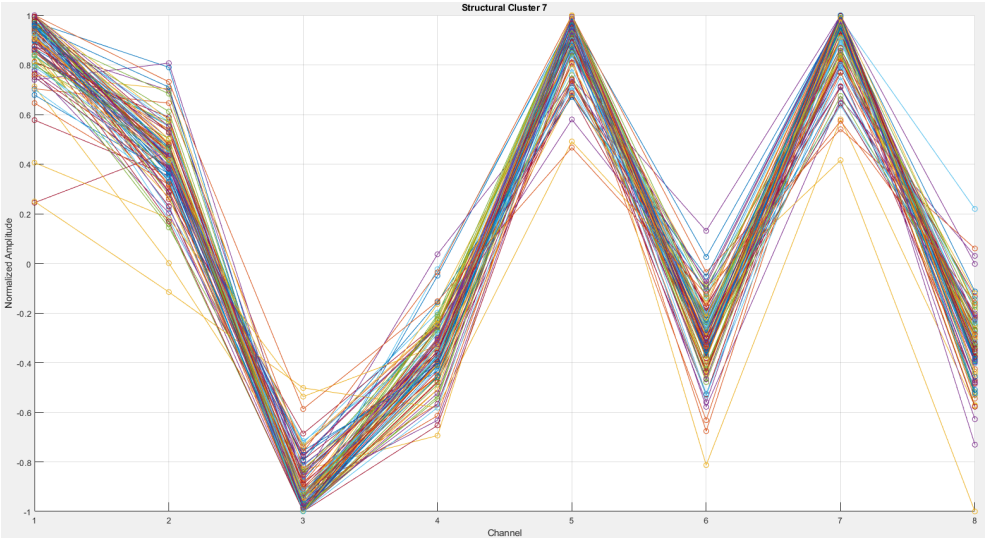


Figure 2.8: Structural cluster found in the ERA dataset. Samples start at 34 Hz in test 1 and drop to 29 Hz at test 12

2.5 Machine learning Framework

2.5.1 Resources and Packages Used

The machine learning pipelines for this project were implemented in Python, utilizing several libraries and frameworks to ensure efficient data processing, model training, and model evaluation. The primary deep learning frameworks employed were TensorFlow and Keras, which facilitated the design and training of neural network architectures. Additionally, Scikit-learn was used for preprocessing tasks, such as data normalization and splitting datasets into training, validation, and test sets.

For data manipulation and numerical computations, Pandas and NumPy were extensively utilized, enabling efficient handling of large datasets and performing mathematical operations. Visualization tasks, including plotting training curves and visualizing predictions, were conducted using Matplotlib. Finally, Scipy was employed for signal processing tasks.

This combination of libraries provided a robust and flexible environment for implementing and refining the machine learning pipelines used in this project.

2.6 Machine learning pipelines

In this chapter the supervised machine learning pipelines developed will be presented. In the following figure 2.9 an overview of the models, distinct datasets and labeling schemes is presented.

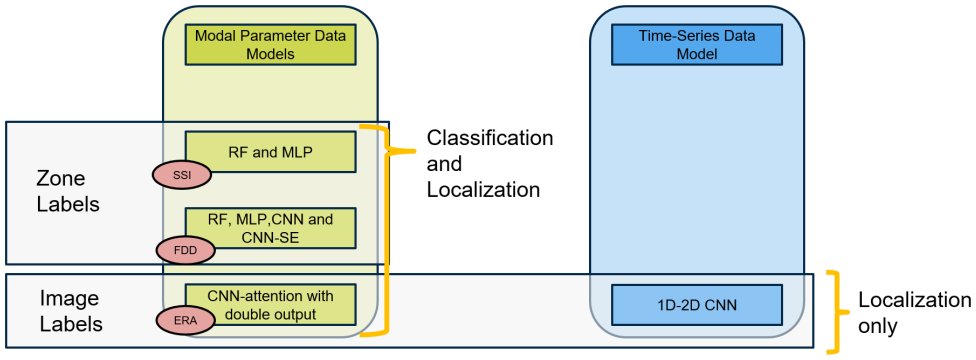


Figure 2.9: Overview of the different machine learning pipelines of this project. **Green**: Modal parameter datasets are used; **Blue**: Accelerometer data is directly used; **Red**: Method used for modal parameter extraction and dataset creation; **Gray**: Indicates Labeling scheme; **Orange**: Clarifies tasks each pipeline aims to perform.

Overall, in the earlier tests using the SSI and FDD datasets, simple neural network (NN) architectures were chosen. However, these NNs tended to overfit, so Random Forest (RF) was used as a comparison, as it typically performs well on small datasets where NNs are prone to overfitting. In later tests using the ERA and time-series datasets, more complex NN architectures were selected, as these datasets provided a larger number of samples.

2.6.1 RF and MLP models using SSI dataset

This following pipeline was the first developed for this project. It uses modal parameters such as natural frequencies, damping ratios and mode shapes extracted using the covariance driven stochastic subspace identification (SSICOV) method.

Data Collection and Augmentation

As mentioned before, the data was extracted using covariance driven stochastic subspace identification. For more details on this method the reader is encouraged to read section 2.
We load the following data:

1. *Natural frequencies*
2. *Damping ratios*
3. *Mode shapes*

Given the limited size of the dataset, we performed data augmentation by introducing 10% Gaussian noise (added to the accelerometer data). For each unique test number and excitation pair, we generated five additional noise-augmented data files.

Reference Mode Shapes

When creating the SSI script, each mode shape found was plotted (2D and 3D plots). In a subsequent step, these mode shapes were visually inspected to identify common, recurring shapes across tests and excitations. These recurring shapes were noted and saved as a selection for the healthy states (i.e., the test data before any damage was detected, specifically from Test 1 to Test 3).
Using these selected mode shapes and the pairwise MAC scores (Modal assurance criterion) representative mode shapes where selected and denoted as Mode A, Mode B, Mode C and Mode D.
The process for creating the `final_reference_modes.csv` can be summarized as follows:

- Visually inspect the exported 2D and 3D representations of mode shapes from healthy-state tests (Test 1 to Test 3) to identify recurring patterns.
- Record the selected mode shapes and their associated test, excitation, and mode numbers in a `Selection.csv` file.
- Compute pairwise MAC scores between the pre-selected mode shapes to quantify their similarity.
- Select the mode shape with the largest *median MAC score* for each mode type as the representative.

Feature Engineering

We compute the MAC between each mode shape and the previously defined set of reference modes. If the highest MAC value exceeds a specified threshold of 0.7, the mode shape is assigned to that reference cluster; otherwise, it is labeled as `Noise`. These assignments are stored as an additional categorical feature named `cluster`. Each feature vector includes:

- **Natural Frequency**
- **Damping ratio**
- **The cluster assignment of each mode**: (e.g. `Mode A`, `Mode B`, `Noise`), subsequently converted into dummy variables using one-hot-encoding.
- **mode shape vector**

After feature extraction, we merge these features with the multi-zone damage labels (the code runs on both the 10 zone labels and the 20 zone labels). All numeric features are standardized (`StandardScaler`) prior to model training.

Data Splitting

We employ a *GroupShuffleSplit* strategy to prevent data leakage. Specifically, each `(Test_ID, Excitation)` combination is treated as a single group, ensuring that all samples from the same CSV file go entirely to either the training or test set. 80% of the data is assigned into the training split and the remaining 20% is assigned to the validation split.

Random Forest Model

The first model is a **Multi-Output Random Forest** (RF) created by wrapping a standard `RandomForestClassifier` in a `MultiOutputClassifier` object. Each zone is treated as a separate binary classification target, allowing the Random Forest to produce class probabilities for each zone. The number of estimators is set to 200, and the maximum depth is set to 15.

MLP Model

In parallel, we train a simple **Multi-Layer Perceptron** (MLP). Each sample is passed to the network as a single vector of standardized features, and the network outputs a multi-label prediction (one neuron per damage zone). The MLP architecture is as follows:

1. **Input Layer**: Accepting a flat 1D feature array.
2. **Dense Layer**: A fully connected layer with 128 neurons. Uses ReLu activation which introduces non-linearity and is regularized using L2 to avoid overfitting.
3. **Dropout Layer**: Drops a specified percentage of neurons during training to avoid over reliance on just some neurons (in this case 30%).
4. **Output Layer**: A fully connected layer with either 10 or 20 neurons, depending on which zone label set is used. A sigmoid activation function is used.

Training uses the `Adam` optimizer and `binary_crossentropy` loss. We apply early stopping to mitigate overfitting.

Evaluation Metrics

We measure performance for the two tasks of damage classification and damage localization:

- **Zone-Level Multi-Label Classification**: Each zone is considered a separate binary label, yielding metrics like zone-level `accuracy`, `precision`, `recall`, and `F1-score`.
- **Binary Damage Detection**: If any zone in a sample is predicted to be damaged, we classify the entire structure as damaged. We evaluate this binary decision via `accuracy`, `precision`, `recall`, and `F1-score`.

The results of this pipelines are presented in chapter 3.

2.6.2 RF, MLP and CNN models using FDD dataset

This section presents a pipeline for training and evaluating different machine learning models such as Random Forest, MLP, and CNN variants on the FDD dataset, using both 10-zone and 20-zone damage labels.

Data Loading and Preprocessing

Modal parameters such as natural frequencies and mode shapes are extracted using the Frequency Domain Decomposition (FDD) method. For more details on the modal parameter extraction process, readers are encouraged to refer to section 2.3.

We begin by loading the modal parameters from CSV files exported by the FDD script. Each CSV file corresponds to a specific `test id` and `excitation`. The data is organized by `Mode Number`, which represents groups of data belonging to the same mode. These modes are identified based on their similar frequencies and mode shapes, as extracted by the FDD MATLAB script. This script further computes statistical metrics such as the average MAC, median MAC, and MAC standard deviation for the Modes found.
In a second step, reference modes are derived from common modes identified in the healthy state (Tests 1 to 3) using MAC scores. The process of determining these reference modes involves identifying and clustering recurring modes from the healthy tests. In a third step, MAC scores are computed between each mode shape in the dataset and the reference modes.
In addition, zone based damage labels are loaded and used as Labels. These labels provide the ground truth for the classification and localization tasks.

Feature Engineering

The feature vector is comprised of:

- **Test ID**
- **Excitation**
- **Mode Number**
- **Natural frequency**
- **MAC-related mode-specific statistics:** Pre-computed statistics for each mode group (Mode Number), including average MAC, median MAC, and MAC standard deviation.
- **Mode Shape Values**
- **MAC-related statistics against reference modes:** For each sample mode shape, MAC scores are computed against all reference mode shapes. These scores form a vector representing the similarity of the sample to each reference mode. From this vector, the following statistics are extracted:
  - The **highest MAC score**, indicating the closest match to a reference mode.
  - The **average MAC score**, representing overall similarity.
  - The **standard deviation of MAC scores**, indicating variability in similarity.
- **Closest Cluster**: Indicates which reference mode the sample is assigned to. If the highest MAC score against the reference mode is below 0.7 the sample is assigned to the noise cluster.

All numeric columns are standardized using `StandardScaler`, while categorical fields (e.g., `excitation`, `closest_cluster`) are one-hot encoded. This ensures the final feature matrix is fully numeric and suitable for machine learning models.

Data Splitting and Grouping

To avoid data leakage, we employ a *GroupShuffleSplit* grouped by a combination of `(test_id, excitation, mode_number)`. This ensures that the same mode found for a specific test and excitation does not get split between the training and testing sets. We use an 80%/20% split for training and testing data, respectively.

Model Training and Hyperparameter Tuning

Random Forest

A Random Forest wrapped in a `MultiOutputClassifier` addresses multi-label classification (one binary task per zone). We use `RandomizedSearchCV` to find the optimal hyperparameters. The following are the optimal values found:

- **n estimators**: 250 (the number of trees in the forest)
- **max depth**: 30 (the maximum depth of each tree)
- **min samples split**: 6 (the minimum number of samples required to split an internal node)
- **min samples leaf**: 2 (the minimum number of samples required to be at a leaf node)
- **max features**: `log2` (the number of features to consider when looking for the best split)
- **class weight**: `balanced` (adjusts weights inversely proportional to class frequencies in the input data, addressing class imbalance during training)

Neural Network Architectures

We compare three neural network architectures, a simple CNN, CNN with SE and MLP, designed for multi-label classification:

- **Simple CNN**: This 1D Convolutional Neural Network processes each individual feature vector independently, treating it as a sequence of features. Its architecture is:
  - **Input Layer**: Accepts input of shape `(input_shape, 1)`.
  - **Convolutional Layer**: A `Conv1D` layer with 64 filters, kernel size 3 x 3, ReLU activation, and padding set to `'same'` to preserve the sequence length.
  - **Batch Normalization**: Normalizes activations for stable training.
  - **Max Pooling Layer**: A `MaxPooling1D` layer with pool size 2, reducing the sequence length by half.
  - **Flatten Layer**: Converts the output into a flat vector for the Dense layers.
  - **Fully Connected Layer**: A `Dense` layer with 128 neurons and ReLU activation.
  - **Dropout Layer**: Dropout with a rate of 30% to prevent overfitting.
  - **Output Layer**: A `Dense` layer with `output_size` (10 or 20 depending on which label set is used) neurons and sigmoid activation, producing independent outputs for each zone.
- **CNN with Squeeze-and-Excitation (SE) Blocks**: This model extends the Simple CNN by including SE blocks (Squeez and Excitation Blocks), which adaptively reweigh channel-wise feature maps to enhance representational power. Its architecture is:
  - **Input Layer**: Shape `(input_shape, 1)`.

  - **Convolutional Layer**: A `Conv1D` layer with kernel size 3 x 3, ReLU activation, and padding `'same'`.
  - **Batch Normalization**: Normalizes activations.
  - **SE Block**:
    * **Squeezing**: Global Average Pooling computes channel-wise statistics.
    * **Excitation**: Two `Dense` layers reduce then restore dimensionality, followed by sigmoid activation. The computed weights are applied to the original feature map.
  - **Max Pooling Layer**: Similar to Simple CNN.
  - **Flatten Layer**: Converts the output into a flat vector.
  - **Fully Connected Layer**: A `Dense` layer with 64 neurons, ReLU activation, and L2 regularization of 0.001.
  - **Dropout Layer**: Dropout with a rate of 30%.
  - **Output Layer**: Same as fo the Simple CNN (10 or 20 neurons).

- **MLP (Multi-Layer Perceptron)**: This model processes the input directly as a flat vector. Its architecture is:
  - **Input Layer**: Accepts a flat input vector of length `input_dim`.
  - **First Fully Connected Layer**: A `Dense` layer with 128 neurons and ReLU activation.
  - **Dropout Layer**: Dropout with a rate of 30%.
  - **Second Fully Connected Layer**: A `Dense` layer with 64 neurons and ReLU activation.
  - **Dropout Layer**: Another Dropout layer with the same rate.
  - **Output Layer**: A `Dense` layer with `output_size` neurons and sigmoid activation, producing independent probabilities for each zone (again 10 or 20 neurons depending on the label scheme used).

All neural network models are trained using the `Adam` optimizer with a learning rate of 0.001, and the `binary_crossentropy` loss function. Early stopping is applied to monitor validation loss, and class weighting is used to address label imbalance during training.

Evaluation Metrics

We measure:

- **Damage Detection (Binary)**: If at least one zone is predicted as damaged, the entire structure is considered damaged. Accuracy, precision, recall, and F1 capture overall performance on this binary decision.
- **Zone-Level Multi-Label Classification**: Each zone is treated as a separate label, yielding multi-label accuracy, per-zone precision/recall/F1, and ROC-AUC (where zone-labeled probabilities are available).

The results of this pipelines are presented in chapter 3.

2.6.3 CNN-attention model with double output using ERA-Clustered dataset

This pipeline utilizes the ERA dataset, which was initially clustered as described in Section 2.4. The ERA dataset was chosen for clustering because it represented the final attempt at improving the modal parameter dataset and contained the largest number of samples. Feature engineering was then performed to incorporate the following features, selected based on findings from [2] and [3]. These studies highlight that mode shape vectors and their derivatives provide information about damage localization, while natural frequencies serve as indicators of damage presence.

Feature Engineering

The chosen features are:

- **Natural Frequency**
- **Mode Shape Vector**: This vector describes the structural deformation mode. According to [2], it remains relatively consistent under damage and provides spatial context.
- **First and Second Derivatives of the Mode Shape Vector**: These derivatives, highlighted by [2], are crucial for identifying localized damage. The first derivative captures changes in stiffness, while the second derivative reveals curvature, aiding in localizing damage regions.
- **Cluster ID**: This indicates whether a mode is structural or noise, derived from the clustering process described in Section 2.4.
- **Normalized Test ID**: The test number, normalized to a range of 0 to 1, provides temporal context, helping the model understand the progression of damage over time. Normalization ensures that the test number's scale does not disproportionately influence model training while preserving the cumulative nature of damage.

The feature vector is constructed by concatenating these components, providing an integrated representation of spatial, temporal, and structural dynamics.

Loading and Processing of Image Labels

Label images for each test ID are loaded and merged into a single composite image, then padded to preserve aspect ratios. This composite label is converted to HSV color space, where pixels in the red hue range are designated as damaged (value 1), and all other pixels are set to undamaged (value 0). As a result, a binary mask is created that shares the same dimensions as the composite image. To address the sparseness of the label, a Gaussian filter is then applied to smooth the mask, preserving the central damaged areas (value 1) while allowing the boundaries to gradually transition toward 0. This step reduces the penalty for minor misalignments, since an approximate localization of the damage region is sufficient, rather than an exact pixel-level prediction of every crack.

Data Splitting and Sequence Preparation

Data is split chronologically into training, validation, and test sets to prevent temporal data leakage since the bridge accumulates damage over time and this is reflected in the image labels which show not only the damage from the current test but also all previous damage. The idea is to learn damage patterns in the data from the

initial tests and assess the performance of the model only on the test set which is comprised of the latest test sets (test 20 to test 25). Feature vectors are grouped into sequences corresponding to each test ID and padded for consistent sequence lengths.

Feature Scaling

All feature vectors are standardized using `StandardScaler`, ensuring numerical stability during training and enhancing convergence.

Model Architecture

The proposed CNN-Attention model processes sequence-level features and produces two outputs:

- **Classification Output:** Predicts the presence or absence of structural damage and uses hard-coded binary labels for the 23 tests.
- **Heatmap Output:** Generates a 256×768 heatmap localizing damage regions on the structure.

The model consists of three main components:

- **Feature Extraction:**
  - Input: Sequences of shape (max_samples, feature_dim), where `max_samples` is the sequence length and `feature_dim` is the feature dimension.
  - Two 1D convolutional layers:
    - **Layer 1:** 128 filters, kernel size 5 x 5, ReLU activation, L2 regularization.
    - **Layer 2:** 64 filters, kernel size 3 x 3, ReLU activation, L2 regularization.
  - Dropout layer with a rate of 0.4 to reduce overfitting.
- **Attention Mechanism:**
  - A dense layer computes attention scores for each sequence step.
  - Scores are normalized using a softmax function.
  - Weighted sum of features generates a fixed-size feature representation (`set_representation`).
- **Output Branches:**
  - **Classification Branch:**
    - Fully connected layer with 256 neurons, ReLU activation, and L2 regularization.
    - Dropout layer with a rate of 0.4.
    - Output layer: Single neuron with sigmoid activation.
  - **Heatmap Branch:**
    - Latent projection reshaped to $16 \times 16 \times 256$. This step projects the 1D embedding (obtained via the attention mechanism) into a multi-dimensional tensor. By doing so, the network transforms a single vector of features into an "image-like" representation, which then serves as the starting point for spatial upsampling through transposed convolutions.

    - Transposed convolution layers:
        - Gradually upsample to $256 \times 768 \times 1$.
        - Kernel size 3×3, strides 2×2 and 1×3, ReLU activation.
    - Final layer: Sigmoid activation ensures output values in $[0, 1]$ for the output mask.

This architecture combines convolutional layers for feature extraction, an attention mechanism for selective weighting across the sequence, and transposed convolution layers for spatial localization (heat maps).

Training and Optimization

The training process for the double-output model architecture employs two distinct loss functions:

- **Classification Output:** For the binary classification task, we use the standard binary cross-entropy (BCE) loss. This loss measures the difference between the predicted probabilities and the ground truth labels, optimizing the model to correctly classify whether structural damage is present or not.

- **Heatmap Output:** For the heatmap prediction task, we employ a composite loss function that combines weighted BCE and Dice loss. Specifically, the combined loss is defined as:

$$\text{Combined Loss} = \alpha \times \text{Weighted BCE} + (1 - \alpha) \times \text{Dice Loss}$$

  where α is a weighting factor (set to 0.5 in our experiments) that balances the contribution of each component. In this approach, damage pixels (labelled as 1) are assigned a higher weight to address their sparsity in the training and validation data. The Dice loss encourages greater overlap between the predicted and true damage regions, promoting better spatial localization.

For evaluation, we use the **Dice coefficient**, which measures the overlap between the predicted and ground truth masks. Unlike BCE, which focuses on pixel-wise accuracy, the Dice coefficient evaluates the quality of the overlap, making it particularly suitable for assessing localization tasks.
The loss functions and evaluation metric are defined as follows:

Binary Cross-Entropy (BCE) Loss:

$$\text{BCE Loss} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

where:

- N: Total number of pixels in the heatmap.

- y_i: Ground truth label for pixel i ($y_i \in [0, 1]$).

- $\hat{y}_i$: Predicted probability for pixel i ($\hat{y}_i \in [0, 1]$).

Dice Coefficient:

$$\text{Dice Coefficient} = \frac{2\sum_i y_i \hat{y}_i}{\sum_i y_i + \sum_i \hat{y}_i}$$

where:

- y_i: Ground truth value for pixel i ($y_i \in [0, 1]$).
- $\hat{y}_i$: Predicted value for pixel i ($\hat{y}_i \in [0, 1]$).

We should note that the Dice Loss is defined as $1 - \text{Dice Coefficient}$. For the loss function, we use Dice Loss, which we aim to minimize (just like the BCE loss). For the performance metric, we use the Dice Coefficient, which measures the overlap. A perfect overlap would thus correspond to a Dice Coefficient of 1.

The combination of a composite loss function for heatmap optimization and the Dice coefficient for evaluation ensures that the model learns to predict damage effectively while being assessed on how well the predicted and true regions align.
The results of this pipeline are presented in Chapter 3.

2.6.4 1D-2D CNN model using time-series dataset

In this section, the machine learning pipeline for a 1D-2D CNN model is described. The pipeline draws inspiration from the works of Dang et al. [9] and Lin et al. [10], adapting their ideas for the specific needs of this project.

Data Loading and Preprocessing

The accelerometer data, stored in CSV files, is loaded for each test. In order to ensure consistent dimensions, each file is truncated to a uniform length of 12,000 datapoints (corresponding to 60 seconds at a sampling rate of 200 Hz).
Although this process standardizes the dataset, we have encountered difficulties in previous pipelines due to the sparsity of high-damage scenarios, which predominantly occur in the final tests. To address this issue and expose the model to high-damage scenarios, we augment the data of the final test (Test 25). Specifically, we generate five augmented copies of this test by adding 5% Gaussian noise, thereby allowing the network to learn from additional instances of high-level damage. We acknowledge that this practice constitutes a degree of data leakage, as it introduces variations of the final test into the training process. Nevertheless, given the imbalanced nature of our dataset and the scarcity of severe damage examples, we choose this compromise to improve the model's ability to predict these rare but critical damage scenarios. Formally, the augmentation is defined as

$$\mathbf{x}_{\text{aug}} = \mathbf{x} + \eta, \quad \eta \sim \mathcal{N}(0, \sigma^2), \tag{2.20}$$

where $\sigma = 0.05\times$ (5% noise level) and $\mathbf{x}$ denotes the original acceleration data.

Normalization and Filtering

First, a high-pass Butterworth filter is applied to remove low-frequency noise with a cutoff frequency of 10 Hz.
Additionally, the data is globally normalized using the mean and standard deviation computed from the training dataset (to avoid data leakage):

$$\mathbf{x}_{\text{norm}} = \frac{\mathbf{x} - \mu}{\sigma + \epsilon} \tag{2.21}$$

where μ and σ are the mean and standard deviation, respectively, and ϵ is a small constant to avoid division by zero.

Label Processing and Heatmap Creation

Each test's damage labels are loaded, resized, and combined while respecting the aspect ratios. The red regions in the annotated images are extracted to form binary masks by converting the images from RGB to HSV color space and thresholding for red hues. These masks are then combined to form a binary mask, with pixels set to 1 for red regions and 0 elsewhere.
To enhance the learning process, a Gaussian filter is applied to the binary masks, smoothing the edges and thickening the lines. This results in a continuous mask with values between 0 and 1, where higher values indicate regions closer to the damage lines. The smoothed masks are normalized and used as ground truth labels for training, enabling the model to learn a continuous representation of damage. For additional context on the labeling method, please refer to Section 2.2.

Data Splitting

The dataset is chronologically divided into training, validation, and testing sets. The training set comprises the initial tests, while the subsequent tests are used for validation. The final tests form the test set, where predictions are visualized and compared to the ground truth labels. This approach ensures a more reliable evaluation of the model's performance, as the primary goal is to accurately identify the regions containing damage, even if the loss values are relatively high.
Chronological splitting is aimed at prevent data leakage, as the damage labels for each test include all accumulated damage up to that point. However, this leads to the previously mentioned issue of imbalanced damage scenarios, as high-level damage only appears in the last four tests and therefore cannot be used for training. To address this limitation, as mentined before, we augment the training set with data from test 25, which contains high-level damage. Therefore, it is important to acknowledge this limitation when interpreting the results, as the model's ability to generalize to previously unseen data may be negatively impacted.

Model Architecture

The model employs a 1D-2D convolutional neural network (CNN) architecture designed to process time-series data and output 2D heatmaps representing damage probabilities. The architecture consists of three main components: an encoder, a bottleneck, and a decoder.

- **Encoder:** The encoder processes the input time-series data $\mathbf{x}_{\text{input}} \in \mathbb{R}^{T \times F}$, where T is the number of time steps and F is the number of features per time step. It uses 1D convolutional layers to extract temporal features, followed by batch normalization and max-pooling layers to reduce dimensionality:

$$\mathbf{z}_1 = \text{MaxPool1D}(\text{BatchNorm}(\text{ReLU}(\text{Conv1D}(\mathbf{x}_{\text{input}})))) \tag{2.22}$$

  This operation is repeated across two convolutional blocks to further refine feature extraction:

$$\mathbf{z}_2 = \text{MaxPool1D}(\text{BatchNorm}(\text{ReLU}(\text{Conv1D}(\mathbf{z}_1)))) \tag{2.23}$$

- **Bottleneck:** The bottleneck layer compresses the extracted features to a lower-dimensional representation. The output of the final convolutional block is flattened, and dense layers are applied:

$$\mathbf{z}_{\text{flat}} = \text{Flatten}(\mathbf{z}_2) \tag{2.24}$$

$$\mathbf{z}_{\text{bottleneck}} = \text{ReLU}(\text{Dense}(\text{Dropout}(\text{ReLU}(\text{Dense}(\mathbf{z}_{\text{flat}})))))) \tag{2.25}$$

This step allows the model to capture a compact representation of the time-series features.

- **Decoder:** The decoder reshapes and upsamples the bottleneck features to generate 2D heatmaps. The bottleneck output is first reshaped to a 2D representation:

$$\mathbf{z}_{\text{reshaped}} = \text{Reshape}(\mathbf{z}_{\text{bottleneck}}) \tag{2.26}$$

Transposed convolutional layers are then applied to progressively upsample the features:

$$\mathbf{H}_{\text{intermediate}} = \text{Conv2DTranspose}(\text{BatchNorm}(\text{ReLU}(\mathbf{z}_{\text{reshaped}}))) \tag{2.27}$$

This process is repeated across multiple upsampling blocks, ultimately producing the final heatmap:

$$\mathbf{H}_{\text{pred}} = \text{Sigmoid}(\text{Conv2D}(\mathbf{H}_{\text{intermediate}})) \tag{2.28}$$

The output $\mathbf{H}_{\text{pred}} \in \mathbb{R}^{256\times768\times1}$ is a 2D heatmap, where each pixel represents the predicted probability of damage.

This architecture is inspired by the works of Dang et al. [9] and Lin et al. [10]. By combining 1D convolutional encoding and 2D decoding, the model effectively maps temporal features from the accelerometer data to spatial damage predictions.

Loss Functions and Metrics

To effectively optimize the model for the task of damage localization, we employ a **combined loss function** with three components: Mean Squared Error (MSE), Dice loss, and an Intersection-over-Union (IoU) term. The combined loss function is defined as:

$$\mathcal{L} = \lambda_{\text{MSE}} \cdot \text{MSE} \; + \; \lambda_{\text{Dice}} \cdot \text{DiceLoss} \; + \; \lambda_{\text{IoU}} \cdot \text{IoULoss}, \tag{2.29}$$

where λ_{MSE}, λ_{Dice}, and λ_{IoU} are the weighting factors for each component. In our experiments, we set $\lambda_{\text{MSE}} = 0.8$, $\lambda_{\text{Dice}} = 1.2$, and $\lambda_{\text{IoU}} = 0.1$. The individual loss functions are defined as follows:

- **Mean Squared Error (MSE):** Minimizes the squared difference between the predicted heatmap $\mathbf{H}_{\text{pred}}$ and the ground truth heatmap $\mathbf{H}_{\text{true}}$:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\Big(\mathbf{H}_{\text{pred}}[i] - \mathbf{H}_{\text{true}}[i]\Big)^2, \tag{2.30}$$

where N is the total number of pixels.

- **Dice Loss:** Emphasizes the overlap between predicted and true damage regions. Defined as:

$$\text{DiceLoss} = 1 - \frac{2 \cdot \sum(\mathbf{H}_{\text{pred}} \cdot \mathbf{H}_{\text{true}})}{\sum(\mathbf{H}_{\text{pred}} + \mathbf{H}_{\text{true}}) + \epsilon}, \tag{2.31}$$

where ϵ is a small constant to avoid division by zero.

- **IoU Loss:** Measures intersection over union. Let $\text{IoU} = \frac{\text{Intersection}}{\text{Union}}$. Then we define $\text{IoULoss} = 1 - \text{IoU}$. Concretely,

$$\text{I} = \sum\big(\mathbf{H}_{\text{pred}} \cdot \mathbf{H}_{\text{true}}\big), \text{U} = \sum\big(\mathbf{H}_{\text{pred}}\big) + \sum\big(\mathbf{H}_{\text{true}}\big) - \text{I}. \tag{2.32}$$

Additionally, several performance metrics are used to evaluate the model:

- **Dice Coefficient:** Measures the overlap between predicted and true regions, similar to Dice loss.
- **IoU Score:** Provides a measure of how well the predicted regions align with the ground truth.
- **Binary Accuracy:** Assesses the proportion of correctly predicted pixels after applying a threshold.
- **Precision:** Evaluates the proportion of predicted positives that are true positives.
- **Recall:** Measures the proportion of actual positives that are correctly identified.

This combination of loss functions and evaluation metrics ensures that the model learns to generate accurate and well-aligned damage localization heatmaps while providing robust performance assessment. The results of this pipelines are presented in chapter 3.

Chapter 3

Results

3.1 Machine Learning Pipelines

3.1.1 RF and MLP models using SSI dataset

Hyper-parameter selection

RF

- n estimators = 200
- max depth = 15
- random state = 42

MLP

- dropout = 0.3
- learning rate = 0.015
- l2 regularization = 0.0005
- batch size = 16
- epochs = 100 (we use early stopping with 20 epoch of patience)

Results for 10-Zone vs. 20-Zone Labels

The pipeline was executed twice: once with the 10-zone labels and once with the 20-zone labels. We summarize the resulting metrics below:

- For the **10-zone labels**:
  - **Random Forest** yields a **damage-detection** accuracy of 68.31%, precision of 94.02%, recall of 56.29%, and F1 of 70.42%. Its zone-level accuracy is 39.4%.
  - **MLP** achieves a **damage-detection** accuracy of 54.32%, precision of 95.87%, recall of 33.27%, and F1 of 49.39%. The zone-level accuracy is 36.8%.
- For the **20-zone labels**:

– **Random Forest** yields a **damage-detection** accuracy of 62.21%, precision of 94.75%, recall of 46.16%, and F1 of 62.08%. Its zone-level accuracy is 37%.

– **MLP** obtains a **damage-detection** accuracy of 46.78%, precision of 92.77%, recall of 22.31%, and F1 of 35.97%. The zone-level accuracy is 34.7%.

Per-Zone Metrics for 10-Zone Labels

Table 3.1: Per-Zone Performance Metrics for 10-Zone Labels (Random Forest and MLP)

Zone	Model	Precision	Recall	F1-Score
Arch SW	RF	0.94	0.30	0.46
	MLP	0.95	0.22	0.36
Arch SE	RF	0.81	0.66	0.73
	MLP	0.77	0.61	0.68
Arch NW	RF	1.00	0.17	0.29
	MLP	0.98	0.19	0.31
Arch NE	RF	0.88	0.42	0.56
	MLP	0.82	0.55	0.66
Spandrel NW	RF	1.00	0.24	0.39
	MLP	0.99	0.44	0.61
Spandrel NC	RF	0.80	0.32	0.46
	MLP	0.87	0.25	0.39
Spandrel NE	RF	1.00	0.33	0.50
	MLP	1.00	0.38	0.55
Spandrel SW	RF	1.00	0.12	0.21
	MLP	0.90	0.18	0.30
Spandrel SC	RF	0.63	0.81	0.71
	MLP	0.74	0.68	0.71
Spandrel SE	RF	1.00	0.33	0.50
	MLP	1.00	0.38	0.55

The table for the 20 Zone performance metrics can be found in the Appendix A.1.

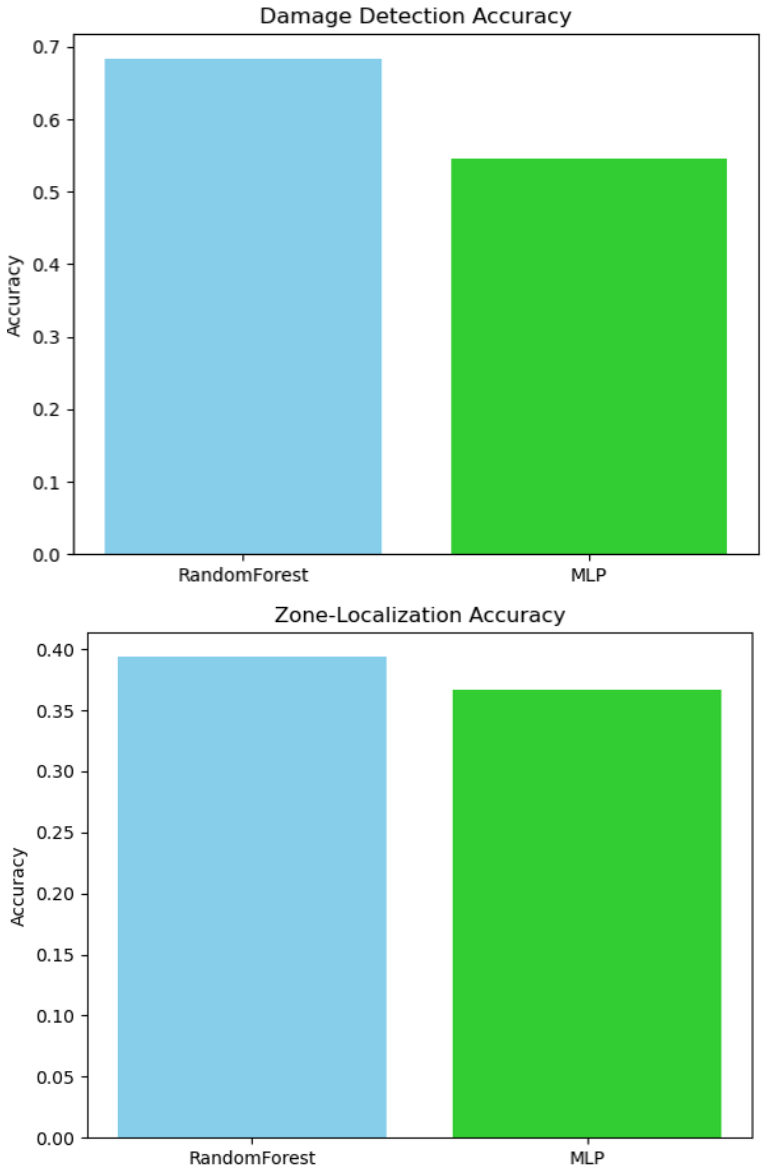


Figure 3.1: Classification and Localization performance of MLP and RF with 10 zones as labels

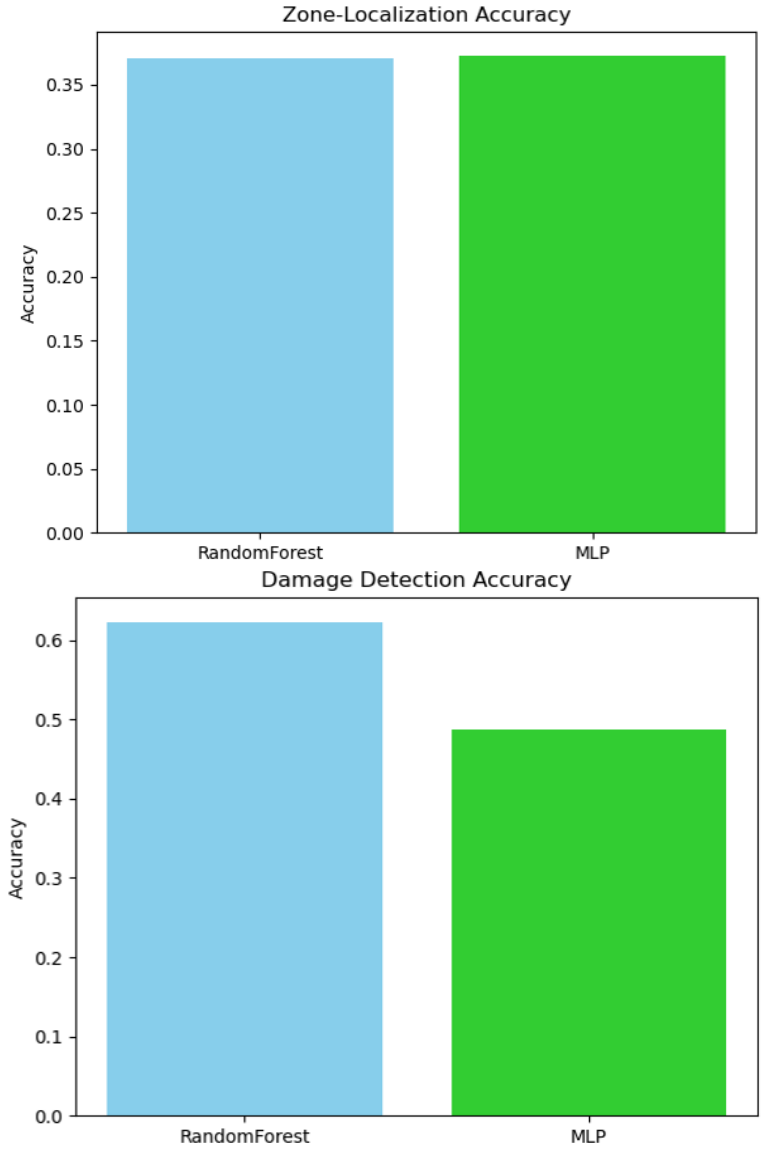


Figure 3.2: Classification and Localization performance of MLP and RF with 20 zones as labels

We observe that, in both configurations (10- and 20-zone labeling), the Random Forest achieves higher damage-detection accuracy than the MLP. However, both models show strong precision and lower recall, indicating that they tend to avoid false positives but often miss certain damaged zones. Moving from 10 zones to 20 zones generally increases the classification difficulty, resulting in slightly lower performance overall.

Overall, these results suggest that the Random Forest model demonstrates more robust performance under both label schemes, whereas the MLP struggles due to the limited amount of data.

3.1.2 RF, MLP, and CNN Models Using FDD Dataset

Hyper-parameter Selection

Random Forest

- n estimators = 250
- max depth = 30
- min samples split = 6
- min samples leaf = 2
- max features = `log2`
- class weight = `balanced`

Simple CNN

- dropout = 0.3
- learning rate = 0.001
- l2 regularization = 0.001
- batch size = 16
- epochs = 50 (with early stopping and 10 epochs of patience)

CNN with Squeeze-and-Excitation (SE) Blocks

- dropout = 0.3
- learning rate = 0.001
- l2 regularization = 0.001
- batch size = 16
- epochs = 50 (with early stopping and 10 epochs of patience)

MLP

- dropout = 0.3
- learning rate = 0.001
- batch size = 16
- epochs = 100 (with early stopping and 10 epochs of patience)

Results for 10-Zone vs. 20-Zone Labels

The pipeline was executed twice, using 10-zone setup and 20-zone setup. We summarize the resulting metrics below:

10 Zones

- **Random Forest**:
  - *Damage Detection*: 92.11% accuracy, 99.14% precision, 90.25% recall, 94.49% F1.
- **Simple CNN**:
  - *Damage Detection*: 73.10% accuracy, 74.74% precision, 96.88% recall, 84.38% F1.
- **CNN with SE**:
  - *Damage Detection*: 74.56% accuracy, 75.11% precision, 98.83% recall, 85.35% F1.
- **MLP**:
  - *Damage Detection*: 74.27% accuracy, 76.04% precision, 95.91% recall, 84.83% F1.

Per-Zone Metrics for 10-Zone Labels

Table 3.2: Per-Zone Performance Metrics (10-Zone Labels) for RF, Simple CNN, CNN with SE, and MLP (Rounded to 2 Decimals)

Zone	Model	Precision	Recall	F1-Score
Arch SW	RF	0.93	0.90	0.91
	Simple CNN	0.51	0.63	0.56
	CNN w/ SE	0.60	0.74	0.66
	MLP	0.49	0.77	0.60
Arch SE	RF	0.91	0.94	0.93
	Simple CNN	0.74	0.86	0.79
	CNN w/ SE	0.71	0.92	0.80
	MLP	0.70	0.82	0.75
Arch NW	RF	0.76	0.61	0.68
	Simple CNN	0.34	0.55	0.42
	CNN w/ SE	0.42	0.62	0.50
	MLP	0.39	0.57	0.46
Arch NE	RF	0.95	0.81	0.88
	Simple CNN	0.74	0.69	0.71
	CNN w/ SE	0.79	0.70	0.74
	MLP	0.83	0.65	0.73
Spandrel NW	RF	1.00	1.00	1.00
	Simple CNN	0.32	0.94	0.47
	CNN w/ SE	0.46	1.00	0.63
	MLP	0.35	0.50	0.41
Spandrel NC	RF	0.90	0.64	0.75
	Simple CNN	0.47	0.54	0.51
	CNN w/ SE	0.49	0.72	0.58
	MLP	0.49	0.64	0.55
Spandrel NE	RF	1.00	1.00	1.00
	Simple CNN	0.00	0.00	0.00
	CNN w/ SE	0.00	0.00	0.00
	MLP	1.00	1.00	1.00
Spandrel SW	RF	1.00	1.00	1.00
	Simple CNN	0.00	0.00	0.00
	CNN w/ SE	0.00	0.00	0.00
	MLP	0.00	0.00	0.00
Spandrel SC	RF	0.99	0.94	0.96
	Simple CNN	0.66	0.80	0.72
	CNN w/ SE	0.73	0.88	0.80
	MLP	0.81	0.72	0.76
Spandrel SE	RF	1.00	1.00	1.00
	Simple CNN	1.00	0.67	0.80
	CNN w/ SE	0.00	0.00	0.00
	MLP	1.00	1.00	1.00

The per-zone metrics for the 20-zone version can be found in the Appendix A.2.

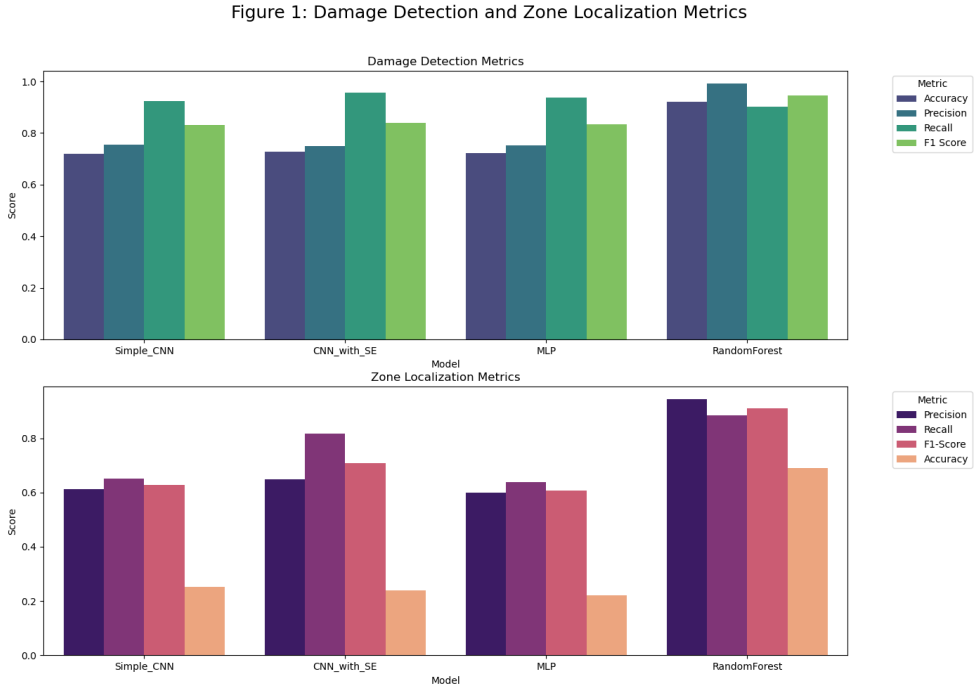


Figure 3.3: Classification and Localization performance of all models (10 zones)

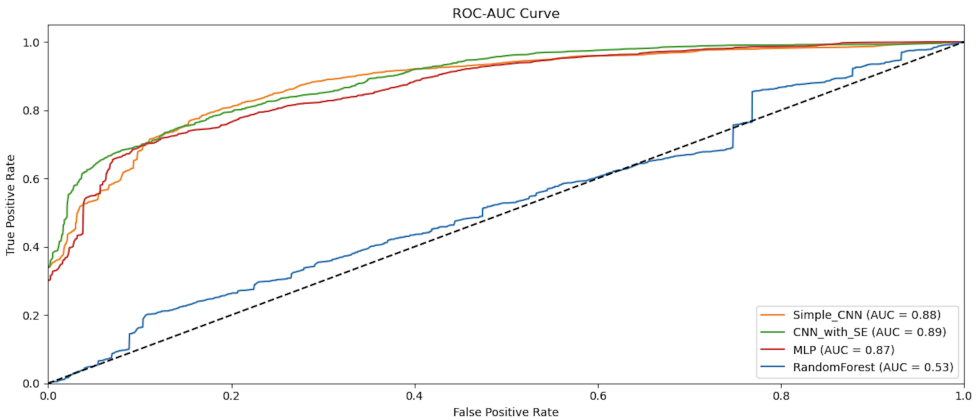


Figure 3.4: ROC-AUC curve for all neural networks

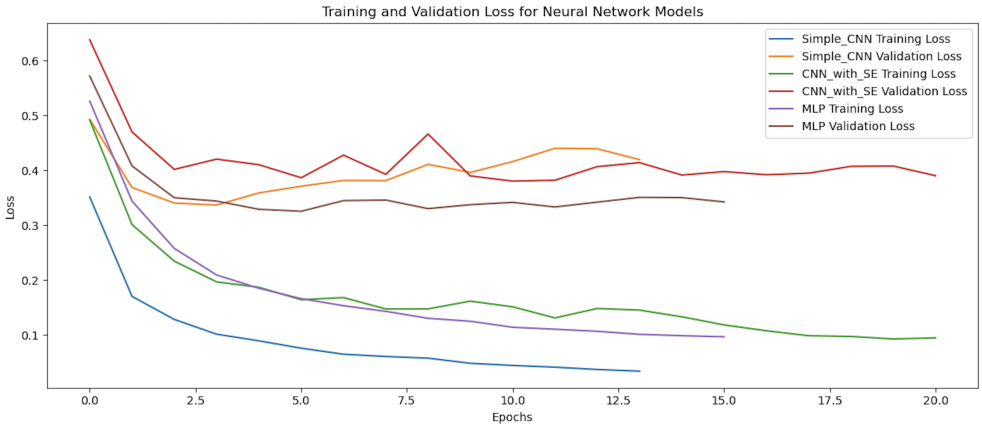


Figure 3.5: Train/Validation loss for all NN models

The Random Forest consistently outperforms neural networks in damage detection. CNN-based models and MLP show competitive recall but struggle with precision and per-zone localization. In both the 10-zone and 20-zone setups, **Random Forest** consistently demonstrates the strongest binary damage-detection performance. Nonetheless, the neural network models (particularly the CNN with SE blocks) deliver higher recall and competitive AUC scores, albeit at the cost of some false positives. All neural network architectures show overfitting tendencies which highlights the lack of data (variety in damage scenarios is small).
Overall, the results underscore how multi-label damage classification becomes more challenging when splitting the structure into more zones. Random Forest remains robust across both labeling schemes, while neural approaches may require additional hyperparameter tuning or more training data to surpass Random Forest's reliability.

3.2 CNN-attention model with double output using ERA-Clustered dataset

The proposed CNN-Attention model continues to demonstrate strong performance in **binary damage classification**. We summarize the results:

- **Training Set**: Classification accuracy of 100%, and heatmap Dice coefficient of 0.3525.
- **Validation Set**: Classification accuracy of 100%, and heatmap Dice coefficient of 0.2430.
- **Test Set**: Classification accuracy of 100%, and heatmap Dice coefficient of 0.1702.

On the test set of four samples (all labeled "damaged"), the model once again achieved perfect classification accuracy, with precision, recall, and F1-scores of 1.00. Despite this seemingly flawless classification result, it is crucial to note that such performance may not generalize beyond this small test split. Since the test set is extremely limited (only four samples, all from the same class—damaged states), the reported 100% accuracy cannot definitively confirm the model's robustness or its ability to handle undamaged examples. A larger and more balanced test set would be essential for a conclusive evaluation of generalization.
It is also worth noting that damage classification is relatively straightforward when analyzing modal parameter changes in the ERA dataset. Clear drops in frequencies,

particularly in the final tests with high-level damage scenarios, indicate structural degradation. Thus, it is reasonable that our model, which was trained on data with small damage variations, achieves perfect damage detection metrics on high-level damage scenarios.
Regarding **damage localization**, we observe that while the training set Dice coefficient for heatmaps reached 0.3525, it dropped to 0.2430 on the validation set and further decreased to 0.1702 on the test set. These low Dice coefficients suggest that the predicted heatmaps only partially overlap with the true damage regions. This is not necessarily a bad outcome, since we mainly need a rough location of the damage and do not require perfect pixel-level accuracy to map every crack. Nevertheless, the Dice coefficient is a useful numerical metric for comparing performance across different splits. The low Dice coefficients may stem from several factors. First, learning precise spatial localization is challenging due to the highly sparse and imbalanced nature of the dataset, where damage pixels are rare compared to non-damage pixels. Second, the relatively small size of the validation and test splits may misrepresent true performance, as the model is evaluated on only a handful of samples, limiting its reliability. Finally, although each test scenario contains a substantial amount of data, the model is only exposed to 23 distinct damage scenarios. This restricted variety reduces the model's ability to generalize across different damage patterns, potentially leading to poorer performance on unseen cases.

The following Figure 3.6 illustrates the predicted heatmaps for selected test samples. Although the model correctly identifies these samples as damaged (with perfect classification), the spatial overlap between predicted damage and ground truth remains moderate to low. A side by side comparison of ground-truth heatmap vs. predicted heatmap can be found in the Appendix A.

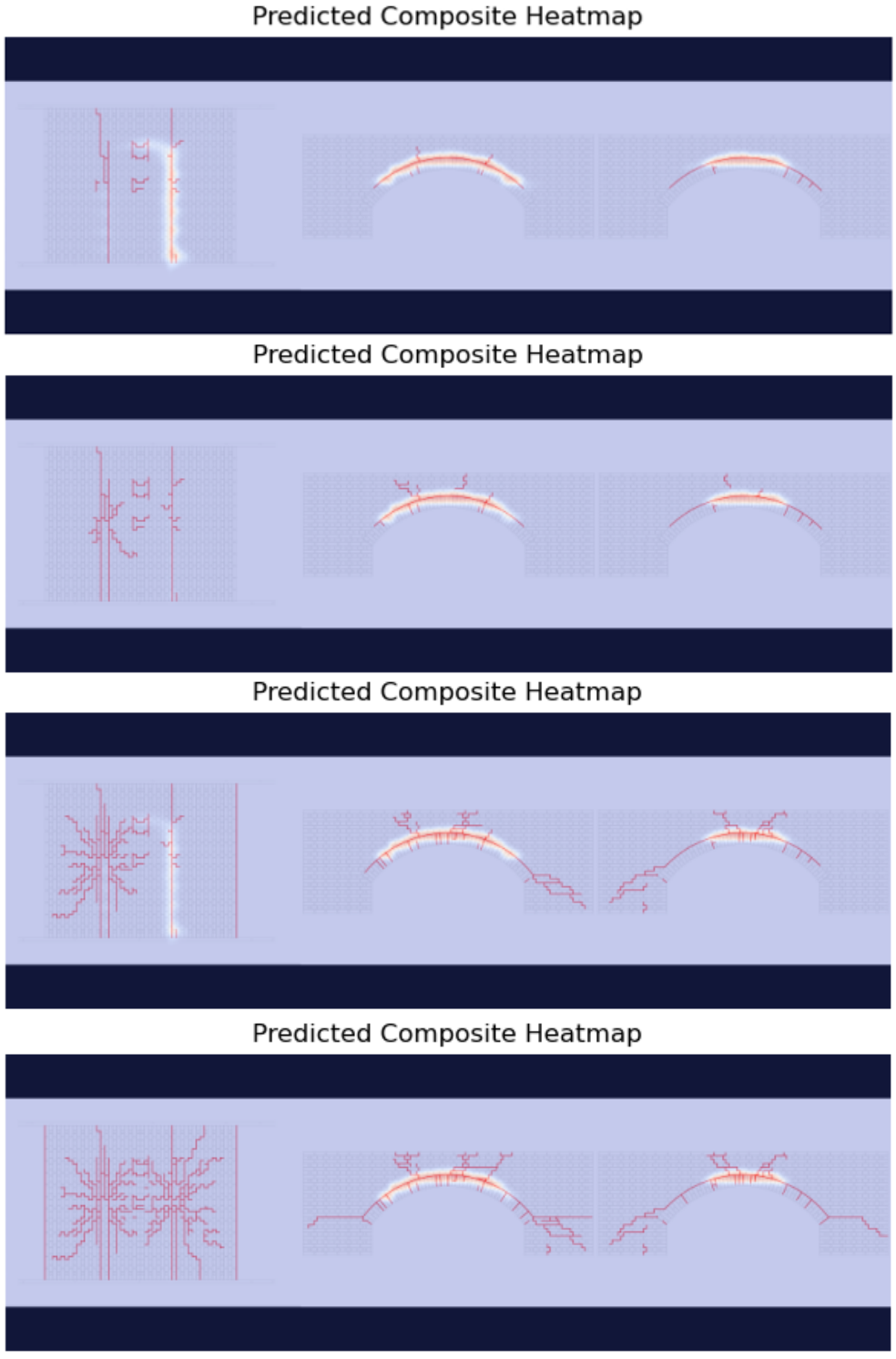


Figure 3.6: Predicted heatmap visualizations for the test set. In order from top to bottom: Test 20, Test 21, Test 22 and Test 25

While the model excels at predicting whether a structure is damaged or not (based on this limited data), the *heatmap localization* remains suboptimal. Moreover, the small size of the validation and test splits means that the apparent *perfect* classi-

fication results may not reflect true real-world generalization performance. These results highlight the difficulty for the model to generalize on high-level damage scenarios.

3.3 1D-2D CNN model using time-series dataset

Hyper-parameter Selection

Training Configuration

- **Learning Rate**: 1×10^{-4}
- **Optimizer**: Adam with gradient clipping (`clipnorm`=1.0)
- **Loss Function**: Combined MSE + Dice + IoU
  ($\lambda_{\text{MSE}} = 0.8$, $\lambda_{\text{Dice}} = 1.2$, $\lambda_{\text{IoU}} = 0.1$)
  - `MSE Loss`: Measures pixel-wise differences.
  - `Dice Loss`: Evaluates overlap between predicted and true heatmaps.
  - `IoU Loss`: Encourages intersection-over-union alignment.
- **Batch Size**: 16
- **Epochs**: 280 (with EarlyStopping and 15 epochs of patience)
- **Callbacks**: EarlyStopping: Monitors validation loss, with a patience of 15 epochs.

In this pipeline, we focus solely on damage localization rather than both damage classification and localization as in previous pipelines. This decision stems from the fact that, to date, we have achieved satisfactory results in damage classification but have encountered challenges in consistently attaining good localization performance, except for the RF model using the FDD dataset.

Results

In the following figures, we present the predicted heatmaps, as well as the training and validation loss and Dice coefficient over the epochs. It is important to note that the displayed results include noise-augmented data from Test 25 to incorporate high-level damage scenarios, which may introduce some bias due to data leakage. This augmentation was necessary because the data in the training split did not contain any high-level damage scenarios.

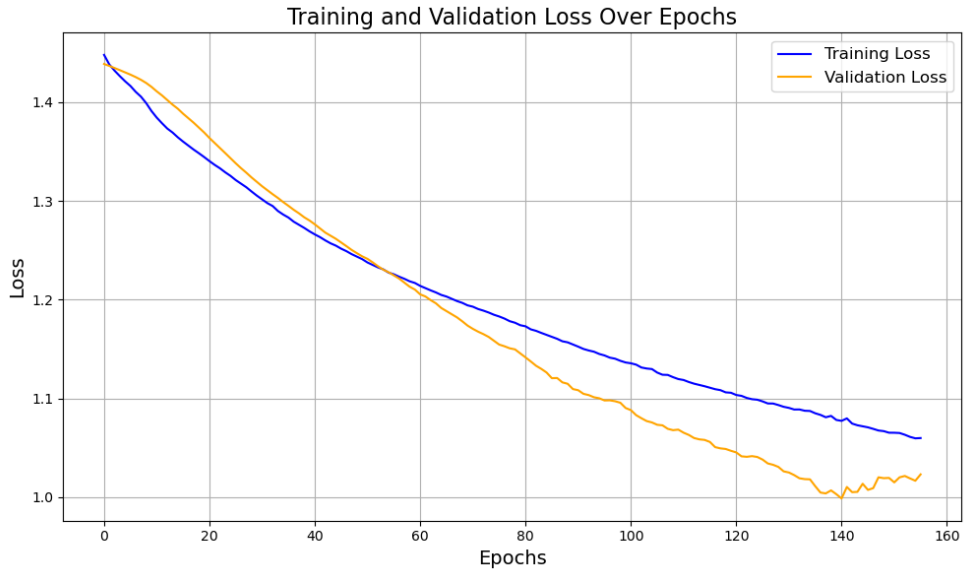


Figure 3.7: Training and Validation loss over epochs.

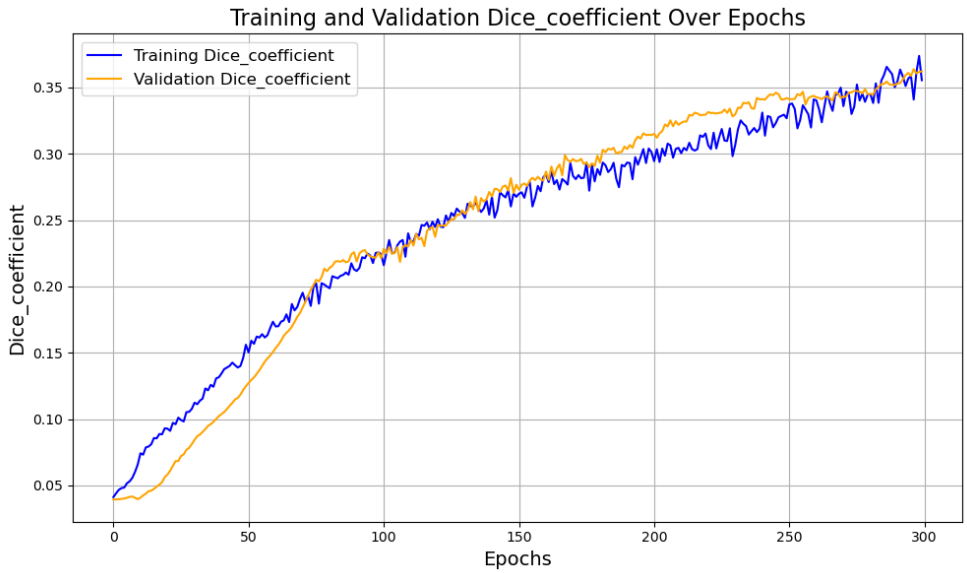


Figure 3.8: Training and Validation Dice coefficient over epochs. A Dice coefficient of 1 indicates perfect overlap.

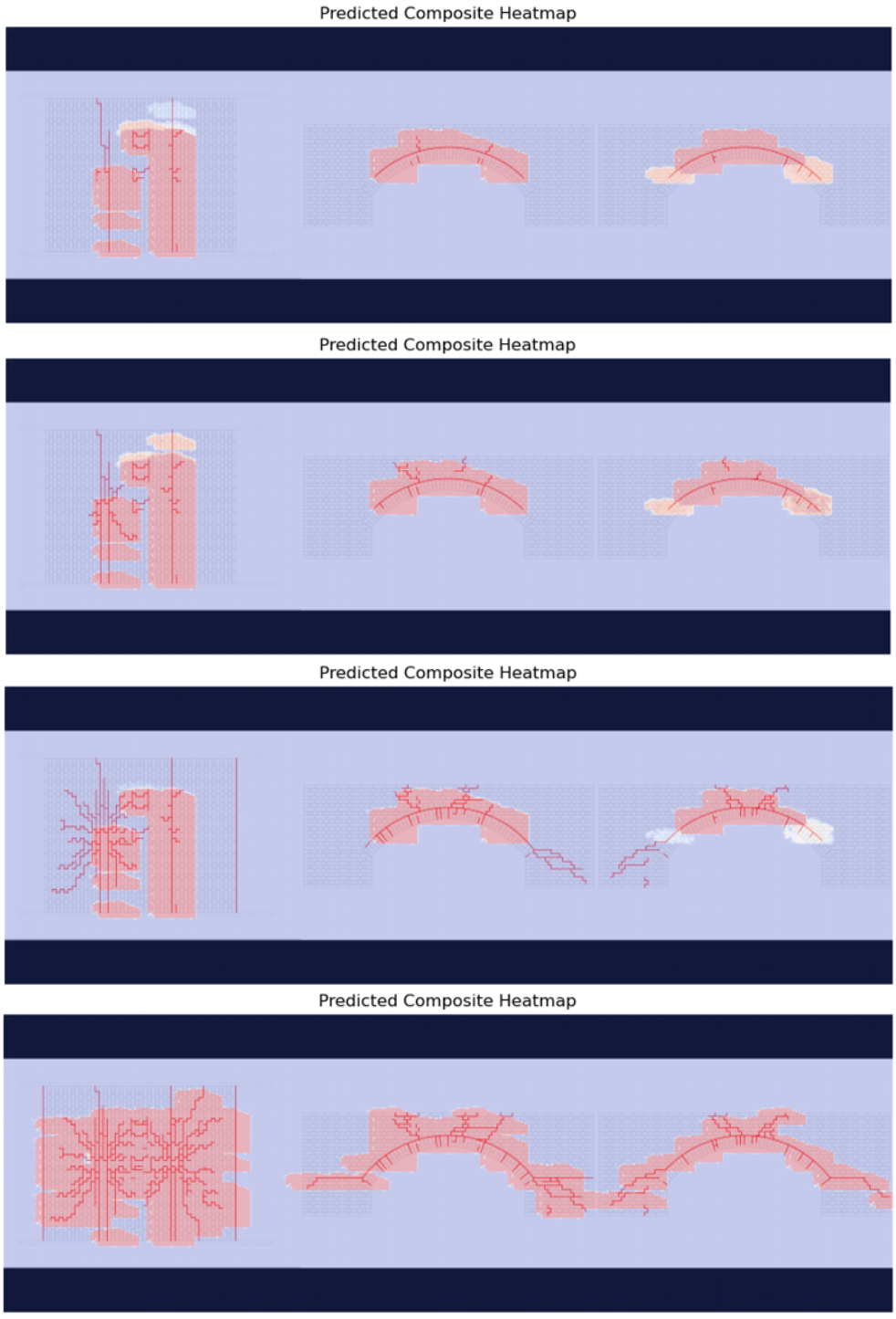


Figure 3.9: Predicted vs. true heatmap visualizations for the test set (from top to bottom Tests 20,21,22 and 25).

In summary, the damage localization performs well, primarily due to the data augmentation applied of test 25. Notably, the model accurately predicted new damage from test 21 to test 22, demonstrating its ability to some extent to learn how changes in data correspond to emerging damage patterns. However, without data augmentation from high-level damage scenarios, the model struggles to produce reliable

results.
It is also important to highlight the significantly higher computational resources required for this time-series model compared to previous models. While earlier models had runtimes of 1 to 3 minutes, this model takes approximately 20 to 30 minutes, depending on the hyperparameter settings, when run on a system equipped with an AMD Ryzen 7 5800X3D CPU, 32 GB DDR4 RAM, and an NVIDIA GeForce RTX 4070 SUPER GPU.
A more precise comparison of computational demands can be derived from the number of trainable parameters in each model. The ERA dataset CNN-attention model contained 4,713,235 trainable parameters, whereas the time-series 1D-2D CNN model had 99,525,297 trainable parameters—an increase by a factor of approximately 21.

Chapter 4

Discussion and Conclusion

4.1 Discussion

This project explores the utilization of machine learning (ML) techniques for the detection and localization of damage in a masonry arch bridge using accelerometer data. Various ML pipelines, including Random Forest (RF), Multi-Layer Perceptron (MLP), and Convolutional Neural Networks (CNN), were evaluated using datasets composed of modal parameters as well as directly from time-series accelerometer data. The findings offer insights into the effectiveness of these approaches and their potential applications in structural health monitoring (SHM).
Initial experiments employed modal parameter datasets derived from data collected in [1] using methods such as Covariance-Driven Stochastic Subspace Identification (SSI-COV), Frequency Domain Decomposition (FDD), and Eigensystem Realisation Algorithm (ERA). Notably, modal parameter extraction in this project proved more challenging than usual, primarily due to difficulties in identifying clear, recurring structural modes. Despite these challenges, the feasibility of classifying a structure as damaged was demonstrated. However, accurately localizing the damage remained difficult given the class imbalance and data sparsity in the labels.

4.1.1 Key Observations and Findings

- **Damage Detection:**
  - When utilizing modal parameter datasets, RF models proved the most reliable for binary damage detection, consistently outperforming neural networks in terms of precision and overall accuracy.
  - Specifically, an RF model applied to the FDD modal parameter dataset with a zone-label-based multiclass classification achieved a damage classification accuracy of **92.11%** and a recall of **90.25%**.
  - A CNN-based model using image-based labeling schemes and the ERA-derived modal parameter dataset also showed promising results for binary classification, although its test set was highly imbalanced, containing only high-level damage scenarios.
- **Damage Localization:**
  - RF models again exhibited superior performance in localizing damage in zone-based labeling schemes, but also demonstrated lower recall in certain zones, highlighting challenges with class imbalance.

  - Image-based CNN models encountered label sparsity and chronological data splitting issues, hindering generalization to high-level damage scenarios.
  - The **time-series CNN** model performed similarly poorly for localization but improved after incorporating noise-augmented samples from later, high-level damage scenarios. Hence, models trained primarily on low-damage scenarios struggled to generalize to more severe damage conditions without this augmentation.

- **Model Performance and Generalization:**
  - Neural networks often underperformed compared to RF models, partly due to the relatively small dataset size and the resulting risk of overfitting.
  - Image-based models showed promising damage classification accuracy but struggled with localization unless augmented data were introduced. This, however is considered data leakage and introduces a bias in terms of generalization on unseen data.

4.2 Conclusion

This work highlights the potential of ML methods for structural health monitoring (SHM), particularly for damage detection in a masonry arch bridge. By applying multiple pipelines ranging from RF and MLP models on modal parameter datasets to 1D–2D CNNs operating directly on time-series accelerometer signals, the following conclusions can be drawn:

- **RF Models for Damage Detection and Localization:** Random Forest models consistently achieved the highest accuracy for binary damage detection and demonstrated robust localization under zone-based labeling schemes. However, lower recall in certain zones suggests that further attention to class imbalance is needed.

- **Neural Networks and Data Imbalance:** Neural network models (both MLP and CNN variants) exhibited lower performance unless data augmentation was introduced. In particular, the image-based CNNs struggled to generalize without exposure to higher-damage samples in the test set. The time-series CNN only improved once noise-augmented samples from Test 25 were added, illustrating the sensitivity of deep learning models to data imbalance.

- **Challenges with Modal Identification:** Extracting modal parameters in this project proved more difficult than anticipated. Identifying stable, recurring structural modes often required extensive manual inspection to address noise, mode mixing, and inconsistent frequency peaks. These complications not only highlight the real-world complexity of working with such data but also partly explain why some localization models did not achieve the expected performance.

- **Chronological Splitting and Data Leakage:** Chronological splitting was necessary to respect the temporal accumulation of damage in the labels but it led to models trained mostly on low-level damage scenarios and tested on high-damage ones. Augmenting the latter with noise improved performance but constitutes data leakage, introducing potential bias.

4.2.1 Limitations and Future Work

While this study provides valuable insights into ML-based SHM approaches, several limitations must be acknowledged:

- **Data Imbalance and Sparsity:** The dataset contained an insufficient variety of severe damage cases, limiting model generalization. Future research should focus on collecting or synthesizing additional data capturing diverse damage scenarios.
- **Unsupervised Learning Potential:** Real-world SHM data often lack labels. Future work should explore unsupervised or semi-supervised approaches (e.g., autoencoders).
- **Advanced Probabilistic Methods:** If collecting more data is not feasible, probabilistic techniques for modeling damage scenarios might provide robust estimates even under limited observations. Such methods could generate synthetic samples to enrich training.
- **Deep Model Requirements:** Deep learning models typically require large-scale training data. Here, the small dataset size contributed to overfitting in most CNN models. Studies with significantly more data and richer damage variations may yield improved CNN performance.

In summary, ML methods, particularly RF and certain CNN variants, have demonstrated promise in detecting and localizing structural damage in a masonry arch bridge. However, the challenges of modal identification, data imbalance, and chronological splitting must be addressed to improve performance and generalization. Future research should prioritize more comprehensive data collection, exploration of unsupervised methods and probabilistic data generation to mitigate bias and enhance real-world applicability.

Acknowledgments

I would like to express my gratitude to the research team at the University of Leeds for their efforts in conducting the study and facilitating data acquisition for this project. I sincerely appreciate their generosity in providing access to the dataset, which was essential for this research.
Furthermore, I extend my heartfelt thanks to my supervisors for their invaluable guidance and feedback throughout this project. In particular, I am deeply grateful to Dr. Charikleia Stoura for providing valuable assistance in developing the modal parameter extraction algorithm using Stochastic Subspace Identification (SSI), Dr. Xudong Jian for developing the modal parameter extraction algorithm using Frequency Domain Decomposition (FDD), and Prof. Dr. Eleni Chatzi for developing the modal parameter extraction algorithm using the Eigensystem Realization Algorithm (ERA). Their expertise and contributions provided the foundation for this work, and their advice and support were instrumental throughout the project.
Lastly, I would like to express my appreciation for the opportunity to collaborate with them and emphasize that it was a privilege and a pleasure to work under their supervision.

Bibliography

[1] B. Liu, D. Kawabe, C.-W. Kim, and V. Sarhosis, "Full life-cycle vibration-based monitoring of a full-scale masonry arch bridge with increasing levels of damage," *Engineering Structures*, vol. 315, p. 118466, 2024.

[2] K. Roy and S. Ray-Chaudhuri, "Fundamental mode shape and its derivatives in structural damage localization," *Journal of Sound and Vibration*, vol. 332, no. 21, pp. 5584–5593, 2013.

[3] O. S. Salawu, "Detection of structural damage through changes in frequency: a review," *Engineering Structures*, vol. 19, no. 9, pp. 718–723, 1997.

[4] E. Cheynet, "Operational modal analysis with automated ssi-cov algorithm," 2020.

[5] R. Brincker and P. Andersen, "Understanding stochastic subspace identification," in *Conference Proceedings : IMAC-XXIV : A Conference Exposition on Structural Dynamics*. Society for Experimental Mechanics, 2006.

[6] R. Brincker, L. Zhang, and P. Andersen, "Modal identification of output-only systems using frequency domain decomposition," *Smart Materials and Structures*, vol. 10, no. 3, pp. 441–445, 2001.

[7] M. Farshchin, "Frequency domain decomposition (fdd)," https://www.mathworks.com/matlabcentral/fileexchange/50988-frequency-domain-decomposition-fdd, 2025.

[8] ETH Zürich, "The eigensystem realization algorithm (era)," https://ethz.ch/content/dam/ethz/special-interest/baug/ibk/structural-mechanics-dam/education/identmeth/ERA_2013.pdf, 2013, accessed: December 2024.

[9] H. V. Dang, M. Raza, T. V. Nguyen, T. Bui-Tien, and H. X. Nguyen, "Deep learning-based detection of structural damage using time-series data," *Structure and Infrastructure Engineering*, vol. 17, no. 11, pp. 1474–1493, 2021.

[10] Y.-z. Lin, Z.-h. Nie, and H.-w. Ma, "Structural damage detection with automatic feature-extraction through deep learning," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 12, pp. 1025–1046, 2017.

Appendix A

Appendix

A.1 Per-Zone Metrics for 20-Zone Labels with SSI Dataset

Table A.1: Per-Zone Performance Metrics for 20-Zone Labels (Random Forest and MLP)

Zone	Model	Precision	Recall	F1-Score
Arch SW	RF	0.94	0.30	0.46
	MLP	0.99	0.16	0.28
Arch SE	RF	0.81	0.66	0.73
	MLP	0.81	0.48	0.60
Arch NW	RF	1.00	0.17	0.29
	MLP	1.00	0.18	0.31
Arch NE	RF	0.88	0.42	0.56
	MLP	0.84	0.50	0.63
Spandrel NW	RF	1.00	0.24	0.39
	MLP	1.00	0.53	0.69
Spandrel NCET	RF	1.00	0.38	0.55
	MLP	0.97	0.62	0.75
Spandrel NCT	RF	1.00	0.38	0.55
	MLP	0.97	0.62	0.75
Spandrel NCWT	RF	1.00	0.24	0.39
	MLP	1.00	0.53	0.69
Spandrel NCEA	RF	0.75	0.30	0.43
	MLP	0.95	0.21	0.35
Spandrel NCA	RF	0.84	0.16	0.26
	MLP	1.00	0.30	0.47
Spandrel NCWA	RF	0.74	0.13	0.21
	MLP	0.96	0.07	0.12
Spandrel NE	RF	1.00	0.33	0.50
	MLP	1.00	0.29	0.45
Spandrel SW	RF	1.00	0.12	0.21
	MLP	1.00	0.41	0.58
Spandrel SCWT	RF	1.00	0.33	0.50
	MLP	1.00	0.29	0.45
Spandrel SCT	RF	0.94	0.43	0.60

(Continued on next page)

(Continued from previous page)

Zone	Model	Precision	Recall	F1-Score
	MLP	0.96	0.62	0.75
Spandrel SCET	RF	1.00	0.12	0.21
	MLP	1.00	0.41	0.58
Spandrel SCWA	RF	0.57	0.51	0.54
	MLP	1.00	0.18	0.30
Spandrel SCA	RF	0.91	0.39	0.55
	MLP	1.00	0.38	0.55
Spandrel SCEA	RF	1.00	0.25	0.40
	MLP	0.00	0.00	0.00
Spandrel SE	RF	1.00	0.33	0.50
	MLP	1.00	0.29	0.45

A.2 Charts and table for the 20 zone approach using the FDD dataset

Figure 1: Damage Detection and Zone Localization Metrics

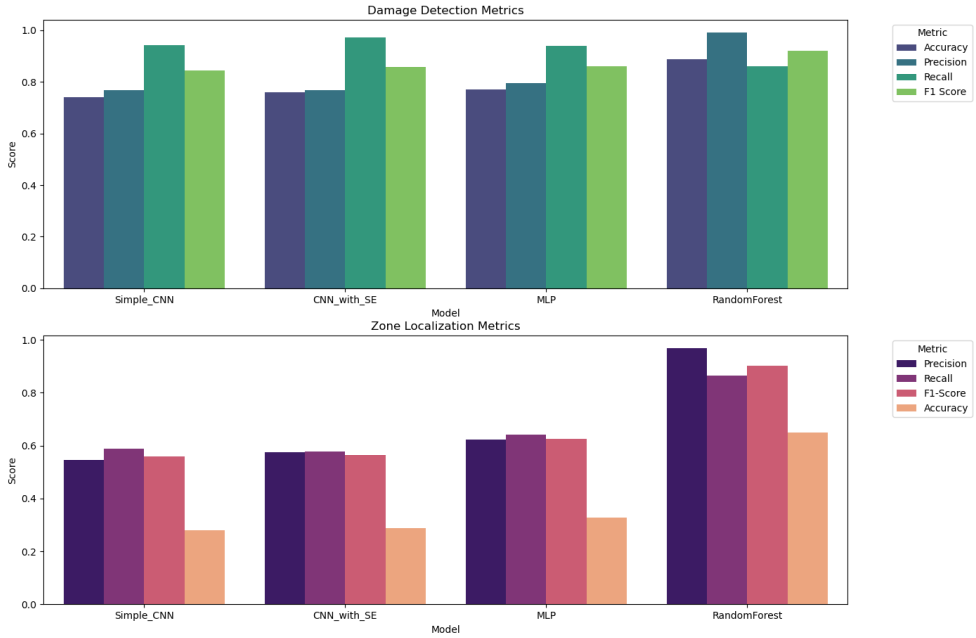


Figure A.1: Classification and Localization performance of all models in comparison

Figure 2: ROC-AUC Curve and Training vs Validation Loss

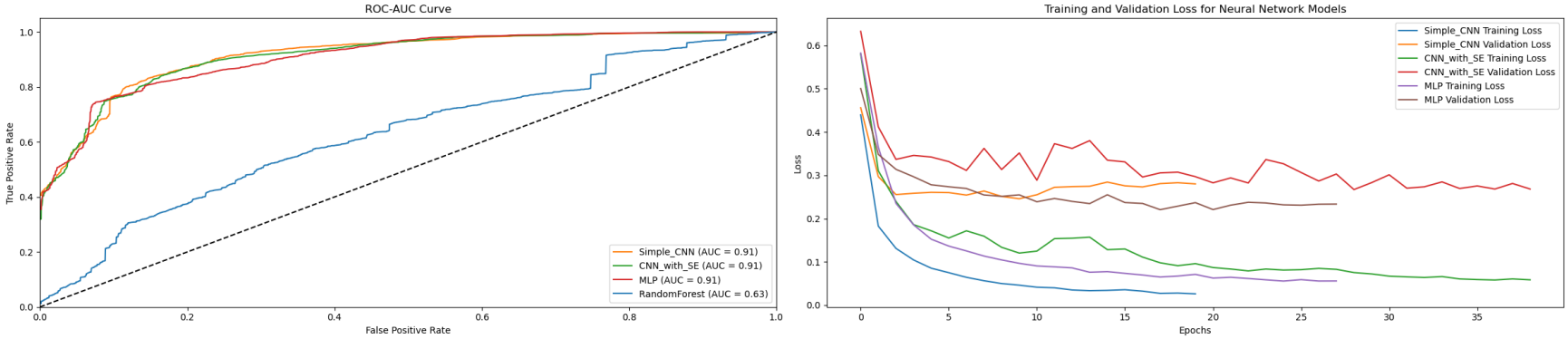


Figure A.2: ROC-AUC curve and Train/Validation loss for all NN

Figure 3: Per-Zone Performance Metrics

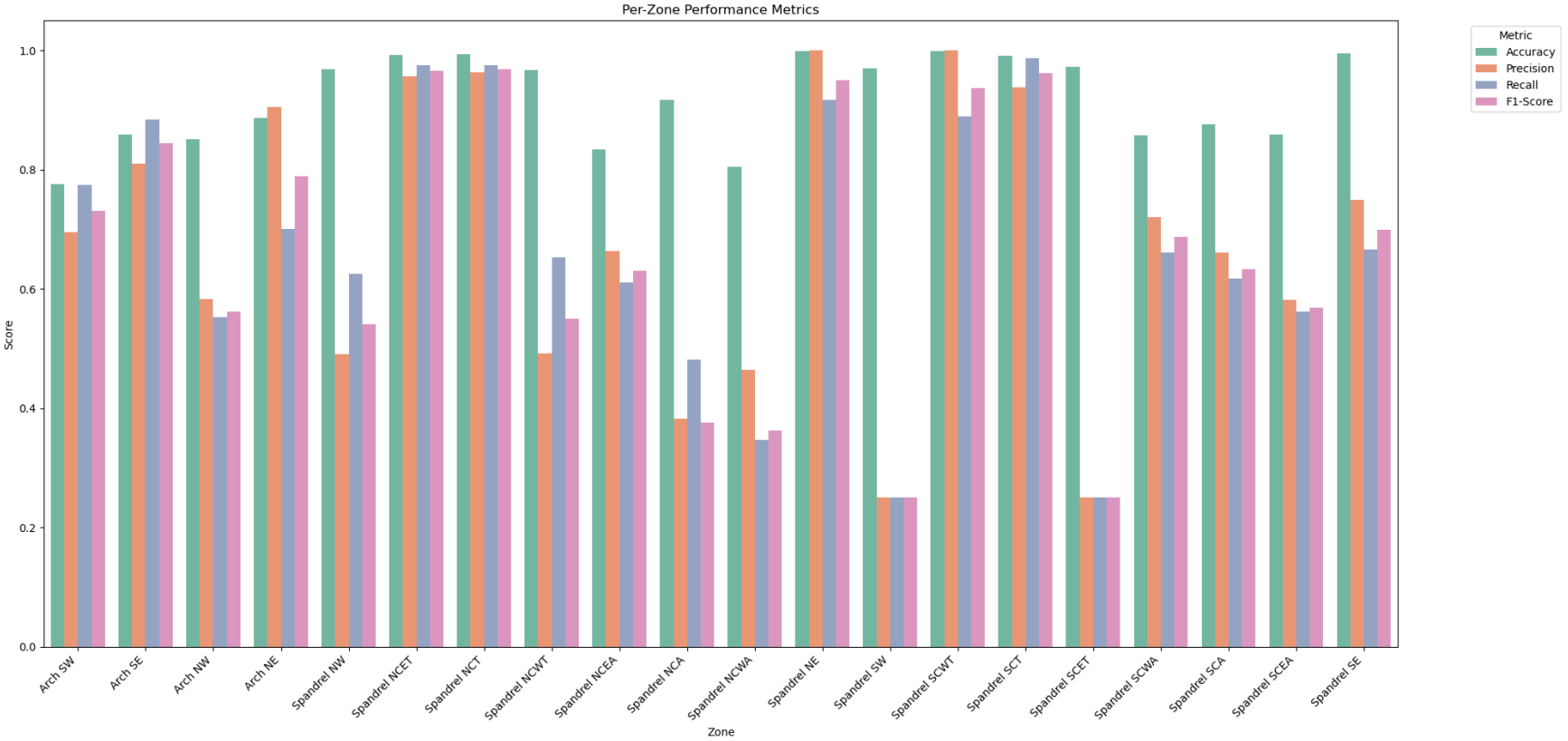


Figure A.3: Per Zone metrics for the RF model

Table A.2: Per-Zone Performance Metrics for 20-Zone Labels (Random Forest and Neural Networks)

Zone	Model	Precision	Recall	F1-Score
Arch SW	RF	0.93	0.90	0.91
	Simple CNN	0.52	0.64	0.57
	CNN with SE	0.63	0.75	0.68
	MLP	0.59	0.79	0.68
Arch SE	RF	0.91	0.94	0.93
	Simple CNN	0.71	0.85	0.77
	CNN with SE	0.73	0.85	0.78
	MLP	0.74	0.87	0.80
Arch NW	RF	0.76	0.61	0.68
	Simple CNN	0.55	0.52	0.54
	CNN with SE	0.50	0.47	0.48
	MLP	0.73	0.58	0.65
Arch NE	RF	0.95	0.81	0.88
	Simple CNN	0.89	0.65	0.75
	CNN with SE	0.90	0.64	0.75
	MLP	0.93	0.64	0.76
Spandrel NW	RF	1.00	1.00	1.00
	Simple CNN	0.40	1.00	0.57
	CNN with SE	0.41	0.72	0.52
	MLP	0.41	0.50	0.45
Spandrel NCET	RF	1.00	1.00	1.00
	Simple CNN	0.84	0.88	0.86
	CNN with SE	0.97	0.97	0.97
	MLP	0.94	0.92	0.93
Spandrel NCT	RF	1.00	1.00	1.00
	Simple CNN	0.79	0.97	0.87
	CNN with SE	0.97	0.97	0.97
	MLP	0.95	0.96	0.95
Spandrel NCWT	RF	1.00	1.00	1.00
	Simple CNN	0.48	0.67	0.56
	CNN with SE	0.46	0.89	0.60
	MLP	0.43	0.50	0.46
Spandrel NCEA	RF	0.99	0.71	0.83
	Simple CNN	0.48	0.58	0.53
	CNN with SE	0.57	0.60	0.59
	MLP	0.53	0.58	0.55
Spandrel NCA	RF	1.00	0.59	0.74
	Simple CNN	0.13	0.26	0.18
	CNN with SE	0.14	0.37	0.20
	MLP	0.16	0.30	0.21
Spandrel NCWA	RF	0.88	0.26	0.40
	Simple CNN	0.21	0.21	0.21
	CNN with SE	0.30	0.37	0.33
	MLP	0.35	0.54	0.42
Spandrel NE	RF	1.00	1.00	1.00
	Simple CNN	0.00	0.00	0.00
	CNN with SE	1.00	1.00	1.00
	MLP	1.00	1.00	1.00

Continued on next page

Table A.2 – continued from previous page

Zone	Model	Precision	Recall	F1-Score
Spandrel SW	RF	1.00	1.00	1.00
	Simple CNN	0.00	0.00	0.00
	CNN with SE	0.06	0.11	0.08
	MLP	0.00	0.00	0.00
Spandrel SCWT	RF	1.00	1.00	1.00
	Simple CNN	0.00	0.00	0.00
	CNN with SE	1.00	1.00	1.00
	MLP	1.00	1.00	1.00
Spandrel SCT	RF	1.00	1.00	1.00
	Simple CNN	0.81	0.96	0.88
	CNN with SE	0.95	0.95	0.95
	MLP	0.95	0.91	0.93
Spandrel SCET	RF	1.00	1.00	1.00
	Simple CNN	0.00	0.00	0.00
	CNN with SE	0.11	0.22	0.15
	MLP	0.00	0.00	0.00
Spandrel SCWA	RF	0.95	0.85	0.90
	Simple CNN	0.49	0.44	0.46
	CNN with SE	0.77	0.64	0.70
	MLP	0.63	0.61	0.62
Spandrel SCA	RF	1.00	0.85	0.92
	Simple CNN	0.35	0.26	0.30
	CNN with SE	0.57	0.59	0.58
	MLP	0.61	0.37	0.46
Spandrel SCEA	RF	0.99	0.80	0.88
	Simple CNN	0.38	0.50	0.43
	CNN with SE	0.48	0.54	0.51
	MLP	0.47	0.33	0.39
Spandrel SE	RF	1.00	1.00	1.00
	Simple CNN	1.00	1.00	1.00
	CNN with SE	1.00	1.00	1.00
	MLP	1.00	1.00	1.00

A.3 ERA pipeline visualizations with ground-truth heatmaps

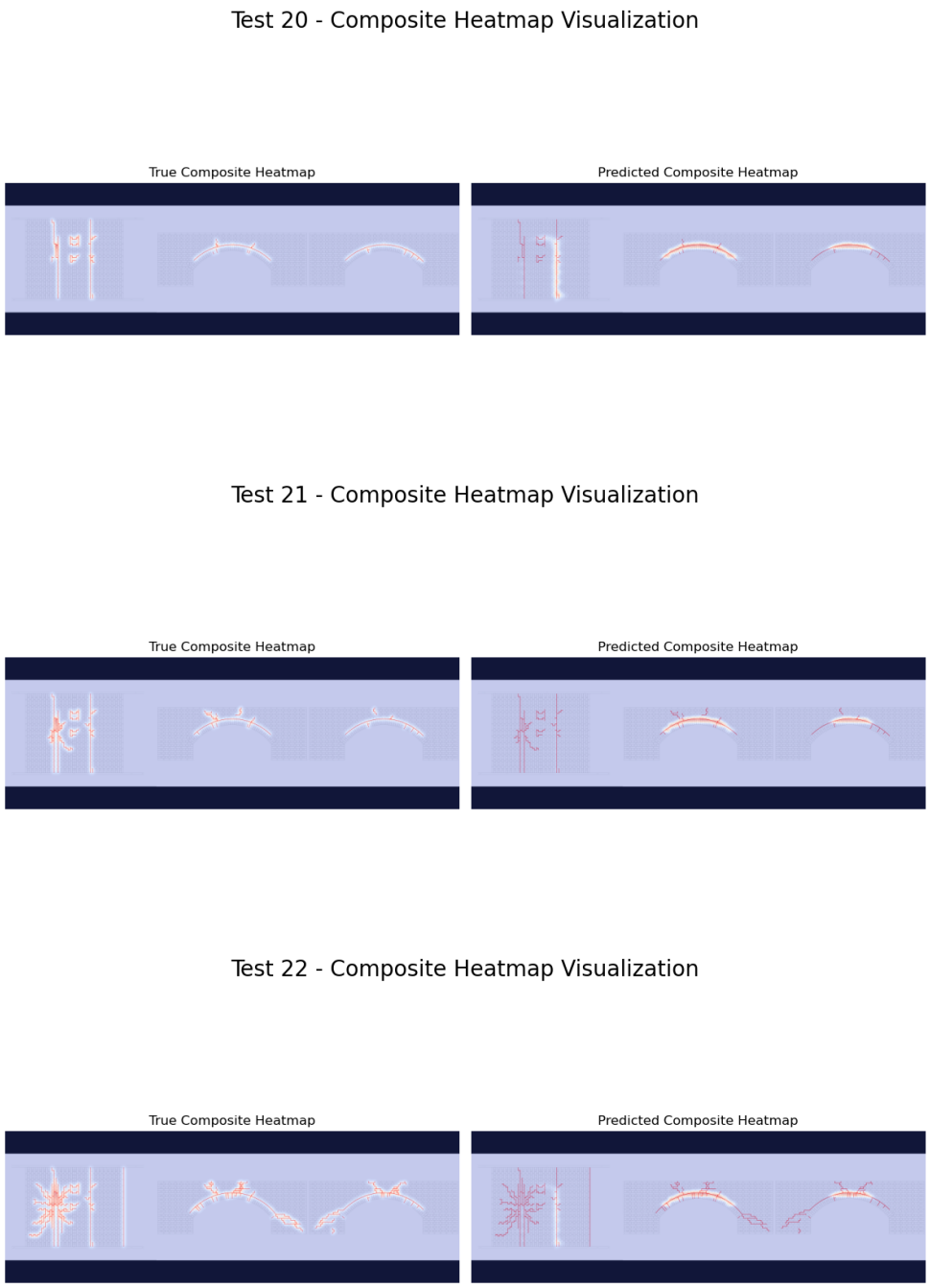


Figure A.4: Predicted vs. true heatmap visualizations for the test set (Tests 20-22).

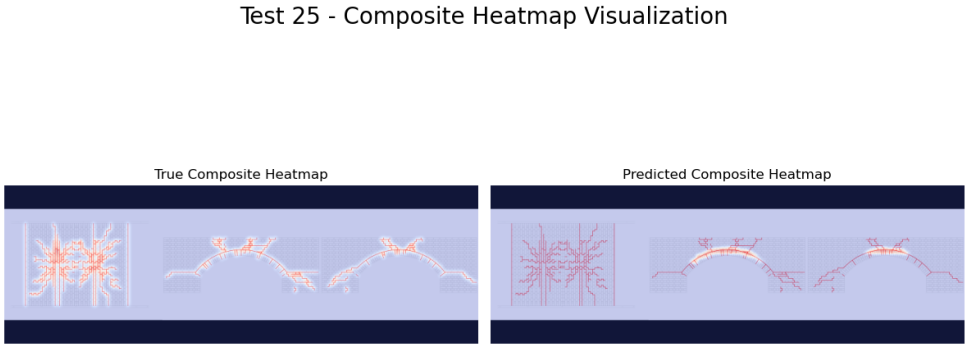


Figure A.5: Predicted vs. true heatmap visualizations for the test set (Test 25).

A.4 Time series pipeline visualizations with ground-truth heatmaps

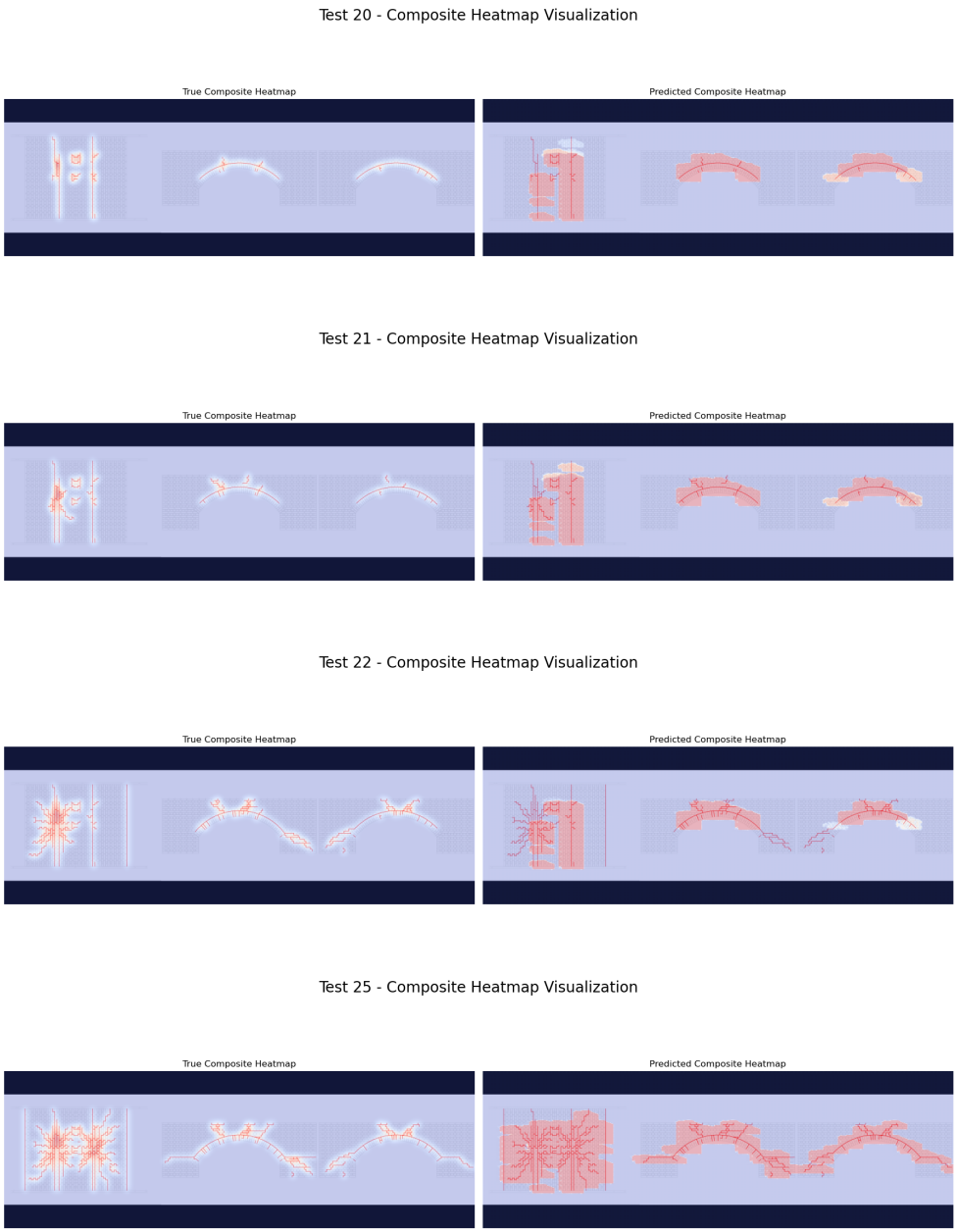


Figure A.6: Predicted vs. true heatmap visualizations for the test set (Tests 20–25).