ModeConv: A Novel Convolution for Distinguishing Anomalous and Normal Structural Behavior

MELANIE SCHALLER, Institut für Informationsverarbeitung, Leibniz Universitat Hannover, Hannover, Germany and Center for Artificial Intelligence and Data Science, Chair for Data Science, University of Würzburg, Wurzburg, Germany
DANIEL SCHLÖR, Center for Artificial Intelligence and Data Science, Chair for Data Science, University of Würzburg, Wurzburg, Germany
ANDREAS HOTHO, Center for Artificial Intelligence and Data Science, Chair for Data Science, University of Würzburg, Wurzburg, Germany

External influences such as traffic and environmental factors induce vibrations in structures, leading to material degradation over time. These vibrations result in cracks due to the material's lack of plasticity compromising structural integrity. Detecting such damage requires the installation of vibration sensors to capture the internal dynamics. However, distinguishing relevant eigenmodes from external noise necessitates the use of Deep Learning models. The detection of changes in eigenmodes can be used to anticipate these shifts in material properties and to discern between normal and anomalous structural behavior. Eigenmodes, representing characteristic vibration patterns, provide insights into structural dynamics and deviations from expected states. Thus, we propose ModeConv[1] to automatically capture and analyze changes in eigenmodes, facilitating effective anomaly detection in structures and material properties. ModeConv automatically learns to set the anomaly threshold based on the highest reconstruction errors found from normal training data, identifying the most pertinent modes for this purpose. After an initial setup, ModeConv efficiently handles the entire analytical process, requiring no further manual adjustments to optimize its performance. The hyperparameters are also optimized automatically using AutoML. In the conducted experiments, ModeConv demonstrates computational efficiency improvements, resulting in reduced runtime for model calculations. The novel ModeConv neural network layer is tailored for spatio-temporal graph neural networks, in which every node represents one sensor. ModeConv employs a novel singular value decomposition based convolutional filter design for complex numbers and leverages modal transformation in lieu of Fourier or Laplace transformations in spectral graph convolutions. We include a mathematical complexity analysis illustrating the runtime reduction.

Additional Key Words and Phrases: Sensor networks, Deep Learning, Spatio-Temporal Graph Neural Network, Engineering, Cyber-Physical Systems

1 Introduction

The condition of materials or structures, such as bridges or buildings, may deteriorate over time due to various factors. To prevent failures and ensure safety, continuous monitoring of their condition is necessary. External

[1]https://github.com/MilanShao/ModeConv

*The authors acknowledge the funding support from the German Ministry for Digital and Transport (BMDV)..
Authors' Contact Information: Melanie Schaller, Institut für Informationsverarbeitung, Leibniz Universitat Hannover, Hannover, Niedersachsen, Germany and Center for Artificial Intelligence and Data Science, Chair for Data Science, University of Würzburg, Wurzburg, Bayern, Germany; e-mail: schaller@tnt.uni-hannover.de; Daniel Schlör, Center for Artificial Intelligence and Data Science, Chair for Data Science, University of Würzburg, Wurzburg, Bayern, Germany; e-mail: schloer@uni-wuerzburg.de; Andreas Hotho, Center for Artificial Intelligence and Data Science, Chair for Data Science, University of Würzburg, Wurzburg, Bayern, Germany; e-mail: hotho@uni-wuerzburg.de.

This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 1550-4867/2026/2-ART
https://doi.org/10.1145/3797951

Just Accepted

forces, as well as environmental effects, can excite structures and materials, causing them to vibrate. These vibrations, combined with factors like deadweight [1, 6, 59], can lead to cracks and damages, ultimately compromising structural integrity.
Traditional methods for condition monitoring, such as visual inspection [2, 12, 21, 40], have limitations in detecting internal damage. To overcome these limitations, the installation of vibration sensors, such as strain gauges or accelerometers [17, 29, 84], allows to capture the vibration behavior of the structure or material. By analyzing the recorded sensor data, it is possible to gain insights into the global vibration behavior and detect changes in the material's condition.
However, the sensor recordings under real-world conditions contain frequencies of external forced vibrations and disturbance variables [30] like environmental conditions, ambient vibrations [47], or external mechanical forces, which overlap with the natural frequencies of the material. These Natural frequencies are inherent frequencies at which a structure naturally vibrates [5, 9, 63] and depend on the structural properties, such as mass, stiffness, and geometry [36, 69]. As the sensor records both, natural frequencies as well as disturbing frequencies, it is difficult to accurately identify [48, 76] and extract the material's characteristic frequencies and modes from the sensor data.

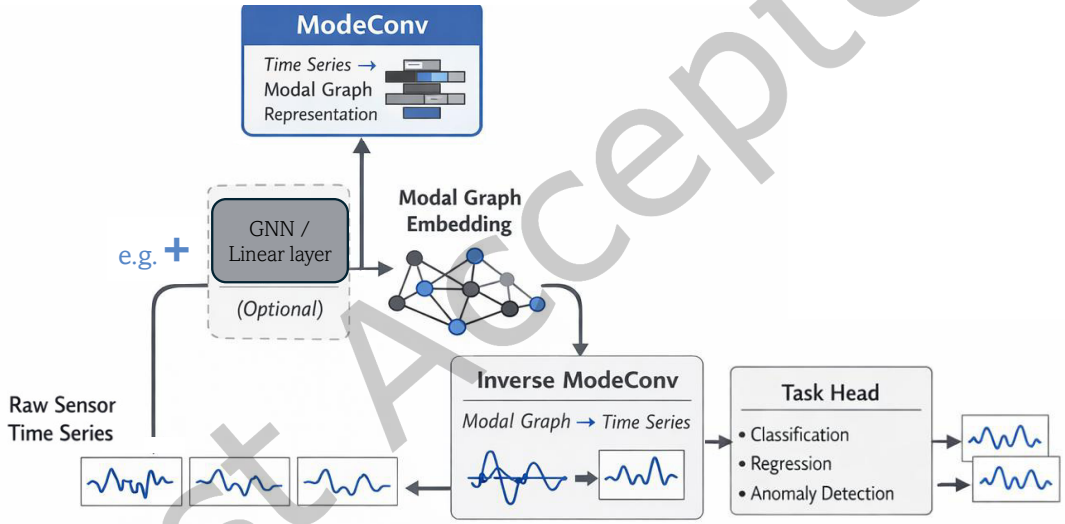

Fig. 1. High-level end-to-end pipeline for ModeConv. Raw multichannel sensor time series can optionally be projected via additional linear, ModeConv or other sorts of graph neural network layers before or after being transformed by the ModeConv layer into a modal graph representation. This graph embedding serves as input to stacked layers for downstream tasks such as classification, regression, or anomaly detection. An optional inverse ModeConv enables reconstruction of time series from modal graphs e.g. in Graph Autoencoder settings.

To address the issue of overlapping frequencies and effectively capture the distinctive frequencies and modes of materials, we utilize a deep learning model with ModeConv layers. These layers can be coupled with any sort of Graph Neural Network (GNN) (see Figure 1), in which every sensor is represented as node and the connections between them as edges, allowing us to effectively capture and separate the relevant information from the sensor recordings (see Figure 2).

Just Accepted

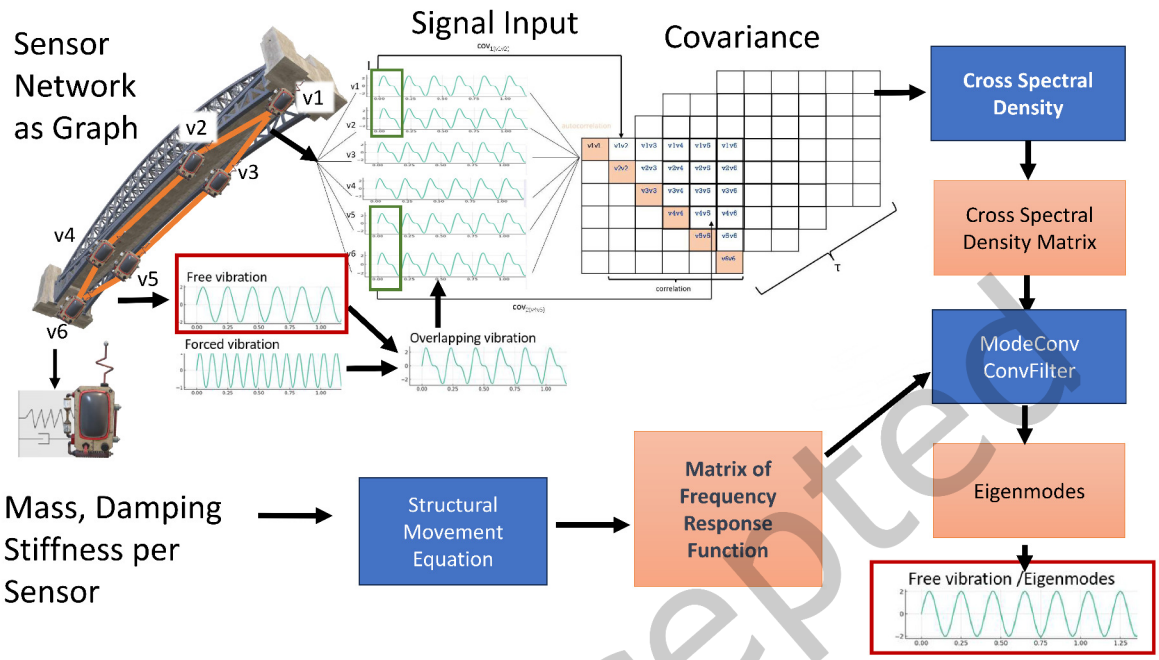

Fig. 2. Visualization of ModeConv Concept: In the upper part starting from the left to the right the installed Sensor Network is showcased as a graph with sensors as nodes and the connections between them as edges; all nodes get the overlapping signal of free vibrations and external vibrations as input signal (showcased with two sinus curves in different frequencies here); these mixed signals are used as input for the covariance matrix; the covariance of signals is further used to calculate the Cross Spectral Density (CSD) and the CSD matrix. In the lower part of the image starting from the left to the right again, the sensornodes are represented as mass points with a certain stiffness and damping, these parameters are used as input for the equation of motion in structural dynamics to calculate the matrix of frequency response functions; Then these two matrices are used together with the CSD matrix as input for the ModeConv Convolutional filter for complex numbers with the imaginary part as first dimension and the real part as second dimension. This filter is designed to automatically learn the Eigenmodes, which are the inherent free vibrations and filters out the external effects, that are not relevant to detect anomalies in the the structural behaviour.

Just Accepted

The objectives of this paper are twofold. Firstly, we aim to develop a method to automatically detect changes in the eigenmodes of structures, allowing for early detection of damages and distinguishing between normal and anomalous structural behaviour. Secondly, we seek to reduce the computational cost of analyzing sensor data without compromising accuracy, paving the way for real-time monitoring applications in the future.
ModeConv, the novel convolutional layer proposed in this work, consists of three major building blocks: (1) The signal block models the covariance [72] of the sensor data, capturing the similarity between different measurements as well as initializing and updating the edge weights [39, 80, 85] of the graph. This block furthermore enables the analysis of the structural behavior by examining the correlation patterns. (2) The Partial Differential Equation (PDE) block covers the physical dynamics of the structure based on the equation of motion [11, 62, 71]. It accurately represents the behavior of the structure by considering factors such as mass and stiffness [12, 28, 69]. By incorporating this physical knowledge, we can gain a deeper understanding of the structural response. (3) The convolutional block combines the information from the signal block and the PDE block to complete the modal

transformation. This process involves extracting the material-typical frequencies and the corresponding natural modes. By aggregating the data through modal transformation, we obtain a comprehensive representation of the global vibration behavior of the structure. Building upon the concept of mode decomposition, we propose a novel graph convolution operation with a complex convolutional filter. In signal processing, signals are often represented as complex numbers [35, 60, 74], where the real part corresponds to the signal's amplitude, and the imaginary part represents the phase shift [70]. By decomposing the signal data into real and imaginary components, these can be passed as separate channels into the two-dimensional convolutional filter of ModeConv. Moreover, ModeConv achieves a reduction in computational cost by efficiently processing only the relevant information. In contrast to previous approaches, that rely on the computation of the symmetric normalized Laplace matrix with Chebyshev polynomials [26], with a time complexity depending on the filter size K, the Singular Value Decomposition (SVD) [43, 73, 77] in ModeConv reduces the matrix multiplication to the number of extracted eigenmodes. This results in significant time savings, especially when large values of K are required. Our software is made available as open source, providing the scientific community with full access to the code and all implementation details. This repository can serve as a foundation for further research.[2]

2 Related Work

The related work is delineated into three principal subsections. The initial division pertains to the model categorization of ModeConv within the framework of geometric deep learning taxonomy. Subsequently, the second subsection elucidates the benchmarking models employed for comparative analysis in the experimental section, while the third subsection delves into the pertinent literature concerning the foundational physical principles underlying the ModeConv approach.

2.1 Model categorization within GNN taxonomy

Graph Convolutional Networks (GCNs) have gained significant attention in the field of graph representation learning [15, 33, 58, 64, 79, 86, 87] and have been widely applied to various domains. Traditional GCNs [52], perform convolutions directly on the graph structure using neighborhood aggregation, but are computational complex. To tackle the computational complexity issue, several approaches have been proposed.

One line of research focuses on approximating graph convolutions using low-rank approximations (e.g. [26]). Spectral-based architectures are built on multiple hidden layers, each one performing spectral graph convolutions defined from a graph signal processing point of view. It provides a notion of frequency and the graph Fourier transform, allowing for filtering in the spectral domain. As a consequence, a graph convolutional layer can be written as a sum of filtered signals followed by an activation function. Each filter is defined in the spectral domain by making use of the eigenvalue decomposition of a graph Laplacian.

Following the proposed taxonomy of Wu et al. [80] to select representative benchmarking strategies, the second line of convolutions can be defined as the line of spatio-temporal graphs, that share the same idea of information propagation with recurrent Graph Neural Networks. The third group consists of hybrid models, that use both.

Our approach differs from previous methods by incorporating a novel complex convolutional filter design based on the Singular Value Decomposition (SVD). Speaking by the taxonomy of Wu et al. [80], we thus formulate a novel group of GNNs, namely the modal graph convolutional networks, that neither represent spectral graph convolutions, nor spatio-temporal graph networks or their coupled versions.

Our modal graph convolutional network, ModeConv, extends the existing GNN taxonomy by introducing a novel group of modal GNNs. Unlike spectral-based approaches that focus on frequency-domain filtering or spatio-temporal methods that emphasize node-level temporal dependencies, ModeConv operates in a hybrid

[2]The open source repository is accessible at: https://github.com/MilanShao/ModeConv.

Just Accepted

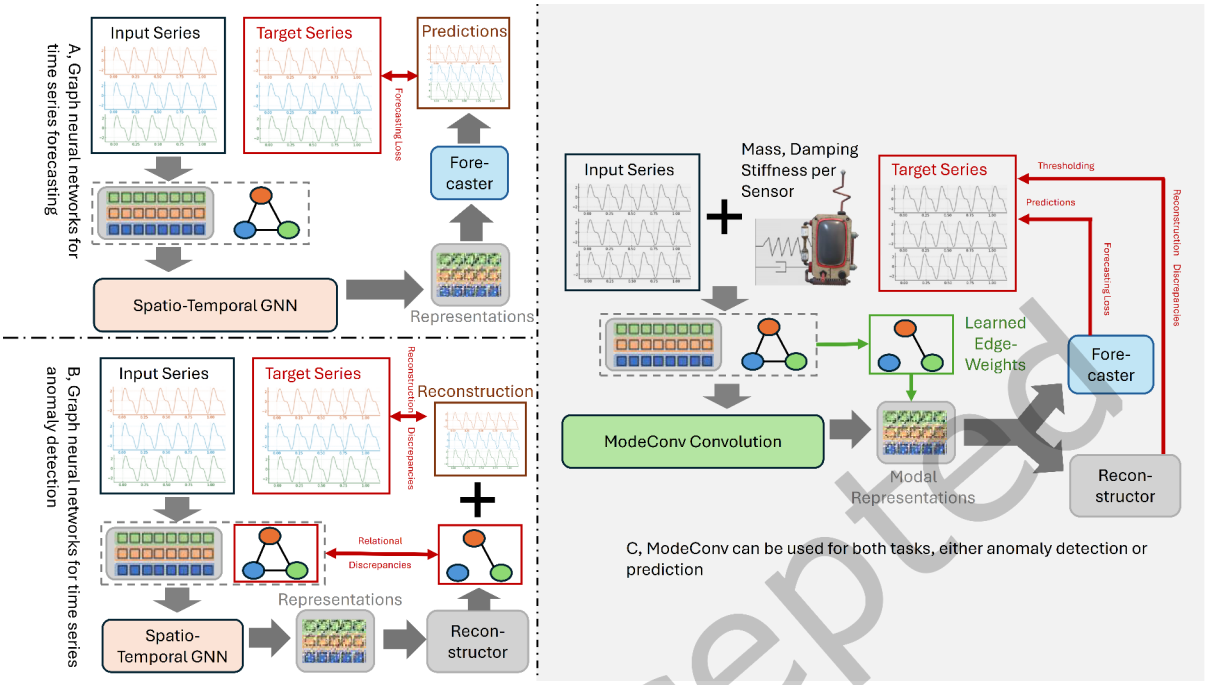

Fig. 3. Overview of Graph Neural Networks (GNNs) for time series analysis. (A) Demonstrates GNNs applied for time series forecasting, where multivariate time series are processed using a Spatio-Temporal GNN to generate representations, which are then used by a forecaster to predict future values. (B) Illustrates the use of GNNs for anomaly detection. Input series are reconstructed using representations learned by the Spatio-Temporal GNN, and discrepancies between the original and reconstructed series indicate anomalies. (C) Highlights the ModeConv module, which integrates sensor-specific physical properties (mass, damping, stiffness) into GNN representations for enhanced performance. ModeConv can be applied to both forecasting and anomaly detection tasks, motivated from [49].

domain where both structural graph features and external domain knowledge are integrated.

To further illustrate these differences, Figure 3 compares the pipelines of traditional GNN-based forecasting and anomaly detection methods with the ModeConv approach.
By categorizing ModeConv as a novel addition to the GNN taxonomy, we aim to bridge the gap between graph representation learning and domain-specific modeling, setting a foundation for further exploration in physical-graph learning paradigms. Due to ModeConv layer's flexibility, it can seamlessly be integrated into various GNN architectures, making it adaptable for tasks ranging from anomaly detection, as demonstrated in our use case as autoencoder layers, to classification tasks.

2.2 Benchmarking models

Spatial dependencies, also known as inter-variable relationships, significantly influence a model's forecasting capability [50]. When dealing with time series data accompanied by graph structures that illustrate the intensity of connections between time series, existing papers commonly utilize (1) spectral Graph Neural Networks (GNNs), (2) spatial GNNs, or (3) a coupled approach incorporating both, to capture and model these spatial dependencies. These methodologies are rooted in the principles of graph signal processing. For our study, we select benchmark

Just Accepted

models based on these grouping. Including their popularity and usage within the research community, we choose at least one model out of these groups, to compare it to the performance of the ModeConv model. Besides these three groups, we also decided to use the proposed GraphCON wrapper [67] as fourth benchmarking section, because it is based on the idea of using nodes as oscillators, which comes closest to the ModeConv idea to model the nodes as positions in a bending wave.

Within the first group of spectral Graph Neural Networks, early papers predominantly employed ChebConv to approximate graph convolution using Chebyshev polynomials. This approach was utilized to model inter-variable dependence. As ChebConv [24] is still a widely-used model for spectral graph convolution, it is often recommended as a reference method for efficient graph convolution and has demonstrated success in numerous applications. StemGNN [13] further introduces spectral-temporal graph neural networks that extract intricate time series patterns by utilizing ChebConv and frequency-domain convolution. Specifically, it captures inter-series correlations and temporal dependencies in the spectral domain using Graph Fourier Transform (GFT) for inter-series correlations and Discrete Fourier Transform (DFT) for temporal dependencies in an end-to-end framework. Although there have been challenges in reproducing the results of StemGNN due to differences in the official implementation, it is still considered a relevant model for spatio-temporal graph networks and thus mentioned here, but the results are excluded from the benchmarking experiments.

Within the second group of spatial GNNs [81], one line of research has been modeling inter-variable dependencies using message passing [37]. The Adaptive Graph Convolutional Recurrent Network (AGCRN) [7] is one of these models, that uses message passing to model inter-variable relations. It incorporates a Node Adaptive Parameter Learning (NAPL) module to capture node-specific patterns and a Data Adaptive Graph Generation (DAGG) module to infer inter-dependencies among different time series automatically. AGCRN outperforms state-of-the-art models on real-world spatio-temporal datasets without pre-defined graphs and spatial connections. Another model of this group is MtGNN [78]. It represents a notable advancement in forecasting models by incorporating graph propagation techniques. This approach allows the integration of neighborhood information from different hops, enabling the learning of high-order relations and substructures within the graph. MtGNN is designed to capture complex spatial and temporal dependencies in time series data. It also enhances temporal convolution by utilizing multiple kernel sizes.

To date, only a limited number of existing methods fall into the category of continuous models. In the realm of factorized methods, STGODE [31] proposes a depiction of graph propagation as a continuous process using a neural ordinary differential equation (NODE) [16]. This approach enables effective characterization of long-range spatial-temporal dependencies, incorporating dilated convolutions along the time axis. On the other hand, for coupled methods, MTGODE (Multivariate Time series with Graph neural Ordinary Differential Equations) [50] generalizes both spatial and temporal modeling processes found in most related works into a single unified process, integrating two NODEs. The MTGODE model was introduced to address key limitations in multivariate time series forecasting. It mitigates three primary challenges posed by existing methods. Firstly, in contrast to discrete neural architectures that interlace individually parameterized spatial and temporal blocks, resulting in discontinuous latent state trajectories and increased forecasting errors, MTGODE presents a continuous model. Secondly, it tackles high complexity by avoiding dedicated designs and redundant parameters found in discrete approaches, thereby reducing computational and memory overheads. Thirdly, it departs from reliance on predefined static graph structures, allowing for dynamic graphs with time-evolving node features and unknown graph structures. MTGODE leverages neural differential equation techniques to complement missing graph topologies and unify spatial and temporal information transfer. Through deeper graph propagation and finely tuned temporal information aggregation, it characterizes stable and precise latent spatial-temporal dynamics. Experimental evaluations on five benchmark time series datasets demonstrate the superior performance of the MTGODE model across various metrics.

The Graph-Coupled Oscillator Networks (GraphCON) [67] framework introduces a novel approach to deep

Just Accepted

learning on graphs. It is built upon discretizations of a second-order system of ordinary differential equations (ODEs), representing a network of nonlinear controlled and damped oscillators coupled via the adjacency structure of the underlying graph. The framework exhibits flexibility by allowing any basic Graph Neural Network (GNN) layer to serve as the coupling function. Through this, a multi-layer deep neural network is constructed via the dynamics of the proposed ODEs. GraphCON addresses the oversmoothing problem often encountered in GNNs by relating it to the stability of steady states in the underlying ODE. The paper assumes that zero-Dirichlet energy steady states are not stable for the proposed ODEs, showcasing the ability of GraphCON to mitigate oversmoothing. Furthermore, the authors demonstrate, that GraphCON addresses the exploding and vanishing gradients problem, facilitating the training of deep multi-layer GNNs. As the GraphCON approach is substantiated through competitive performance compared to state-of-the-art methods across various graph-based learning tasks, it is used as wrapper for benchmarking in this paper.
By choosing the benchmarking models according to these four groups, we can evaluate their performance and compare it on a broader perspective. This comparison helps assess the strengths and weaknesses of different modeling techniques for anomaly detection, as they have varying abilities to capture different aspects of graph data.

2.3 Additional Benchmarking Models

In the last part of the results section, we also offer a comparison against other state-of-the-art deep geometric models for time-series and the MLP baseline in terms of anomaly detection. These models include MLP, TCNAE, VAE, TGCN and STGCN.

The Multilayer Perceptron (MLP) [34] is a type of feedforward neural network characterized by its multiple layers of interconnected neurons. In an MLP, information flows through the network in one direction, from the input layer to the output layer. Each neuron in a layer is connected to every neuron in the subsequent layer, and each connection has an associated weight. The MLP uses activation functions to introduce non-linearity, allowing it to model complex relationships in data. MLPs are widely employed for various tasks, including classification and regression, owing to their capability to learn intricate patterns and representations from input data.

The TCNAE (Temporal Convolutional Network Autoencoder) model [75] is an autoencoder variant designed to capture temporal dependencies in data. It employs a structure of temporal convolutional layers to extract temporal patterns effectively. The autoencoder comprises an encoder that reduces input data to a compact representation and a decoder that aims to reconstruct the original data from this representation. With a convolutional neural network (CNN)-like architecture, TCNAE adeptly captures both spatial and temporal dependencies in sequential data, such as time series or sensor data. The use of convolutional layers enables the model to learn localized patterns while considering the temporal structure of the data. TCNAE finds applications across domains, including signal processing, time series analysis, and tasks where the temporal dimension of the data is critical.

The Variational Autoencoder (VAE) [51] is a generative model based on the autoencoder framework. Unlike traditional autoencoders, VAE incorporates probabilistic elements to model uncertain representations. The encoder transforms input data into a probability distribution in the latent space, determined by mean and standard deviation. The decoder then generates data points from random samples of this distribution. By introducing sampling in the latent space, VAE allows for the generation of more diverse and stochastic outputs compared to traditional autoencoders. VAE is applied in generative modeling, particularly for generating new data points, images, or other complex structures, while accounting for uncertain representations.

The Temporal Graph Convolutional Network (TGCN) [88] is a specialized neural network designed for handling temporal graph-structured data. It extends traditional graph convolutional networks (GCNs) to incorporate temporal dynamics, making it suitable for time-evolving graph datasets. TGCN leverages both spatial and temporal information by applying graph convolutions across both dimensions. This enables the model to capture evolving

Just Accepted

patterns and dependencies in dynamic graph structures, making it particularly effective for tasks involving time-varying relationships. TGCN finds applications in various domains, including social network analysis, traffic forecasting, and any scenario where understanding temporal changes in graph-structured data is essential.

The Spatio-Temporal Graph Convolutional Network (STGCN) [85] is a specialized neural network designed to handle spatio-temporal data with a graph structure. It extends traditional graph convolutional networks (GCNs) to incorporate both spatial and temporal dependencies simultaneously. STGCN is particularly well-suited for tasks involving data that varies across both spatial and temporal dimensions, such as traffic flow prediction, weather forecasting, or video analysis. By leveraging graph convolutions in conjunction with temporal convolutions, STGCN captures complex spatio-temporal patterns, making it effective for understanding and predicting dynamic behaviors in graph-structured data over time.

2.4 Physical Background

The field of civil engineering and structural health monitoring has witnessed significant advancements, particularly in the detection and assessment of structural damage. This section reviews key contributions in civil engineering and structural health monitoring, focusing on the ModeConv approach.

Vibration-based techniques have been extensively explored as non-destructive methods for assessing structural integrity. Dai et al. [20] categorized vibrations into free and forced types, where free vibrations occur when a structure oscillates on its own after displacement, and forced vibrations result from external energy application [71]. Modal analysis has been pivotal in studying vibration characteristics [63]. Salawu [68] emphasized the importance of forced vibration testing, while ambient vibration testing utilizes natural vibrations during operation [56].

The ModeConv approach integrates deep learning with structural knowledge to analyze structural vibrations and detect changes. Statistical and spectral analysis techniques, including the Wiener-Chintschin relation [41], CSD, and covariance, play crucial roles in this integration. In the context of Operational Modal Analysis (OMA) [44, 65], various approaches have been researched to accurately identify modal parameters of dynamic systems under operational conditions. One notable contribution is the work by Araújo and Laier (2014) [4], which proposes a method using singular value decomposition (SVD) of power spectral density transmissibility matrices (PSDTM-SVD). This technique is designed to determine natural frequencies and mode shapes, mitigating the dependence on excitation characteristics that typically affect other methods such as Frequency Domain Decomposition (FDD) [49] or Stochastic Subspace Identification (SSI) [66]. While the PSDTM-SVD method shares the goal of identifying modal parameters, it primarily addresses the challenges inherent in traditional OMA techniques by refining the extraction of these parameters through matrix decompositions. In contrast, the ModeConv approach, introduced in our work, extends beyond mere identification by integrating these modal insights directly into a neural network framework designed for real-time anomaly detection and structural health monitoring.

In the specific task of anomaly detection [10, 14, 42, 61, 83] used in this paper, an anomaly refers to an unusual or atypical behavior or condition observed in the data collected from sensors deployed in a structure. Anomalies are deviations from the expected or normal patterns in the sensor readings. The goal of anomaly detection in the monitoring of structures is to identify these deviations, which may indicate potential structural issues, damage, or abnormal conditions within the monitored system. Anomalies in sensor data could be indicative of structural faults, damage, environmental changes, or other factors that require attention and further investigation to ensure the integrity and safety of the monitored structure.

Just Accepted

The Wiener-Chintschin relation [41] establishes a mathematical connection between Cross Spectral Density (CSD) and the cross-covariance function of a stochastic process. This relation enables ModeConv to analyze cross spectral density, providing insights into the distribution of power across vibration frequencies.

CSD estimation [41] allows ModeConv to estimate the transfer function and coherence between the measured response and assumed excitation, essential for understanding structural behavior. Covariance [72] computation forms the basis of the Signal Block, capturing similarity patterns between signals measured by accelerometers.

Incorporating the physical dynamics of structures is essential for accurate structural health monitoring. The equation of motion, relating structural motion to mass, stiffness, and damping, is fundamental [63]. The PDE Block in ModeConv considers this equation, which is analyzed in the frequency domain using the Fourier transform [18]. This analysis yields the frequency response function, crucial for understanding structural behavior.

The transfer function, derived from the frequency response function through the Laplace transform [28], is crucial for ModeConv. It represents the relationship between input excitation and structural response, aiding in the identification of changes or anomalies.

By incorporating the equation of motion, frequency response function, and transfer function, the PDE Block of ModeConv integrates physical dynamics into the deep learning framework. The PDE Block provides a deeper understanding of the structure's response to external forces and vibrations.

3 Datasets

In this section, we introduce two significant datasets for structural health monitoring (SHM). The first dataset is the *Simulated Smart Bridge dataset*, designed to offer controlled and diverse damage scenarios with varying severity levels. This artificial dataset enables comprehensive training and testing of machine learning algorithms, providing insights into the effects of different types and degrees of damage on sensor readings and network behavior.

The second dataset, the *Luxemburg dataset*, is a real-world dataset that presents realistic damage scenarios observed in actual bridge structures. It incorporates environmental and operational factors, such as temperature and traffic loads, adding layers of complexity that are challenging to replicate in artificial datasets. Analyzing this dataset allows us to understand the behavior of sensor networks in practical situations and validate our methods with authentic, real-world data.

By leveraging both datasets, we aim to address the limitations inherent in each. The artificial dataset may lack the intricacies and uncertainties of real-world scenarios, while the real-world dataset might have fewer damage scenarios under controlled laboratory conditions. The integration of these datasets contributes to a broader understanding of machine learning models in structural health monitoring compared to many existing papers in the domain. The challenge for both datasets is the accurate distinction between external excitations and free vibrations, crucial for precise structural behavior analysis and anomaly detection.

3.1 Simulated Smart Bridge Dataset

The Simulated Smart Bridge dataset was released by the Federal Highway Research Institute (BASt) [32] within the framework of the *Intelligente Brücke* (smart bridge) initiative, a component of the *Testfeld Digitale*

Just Accepted

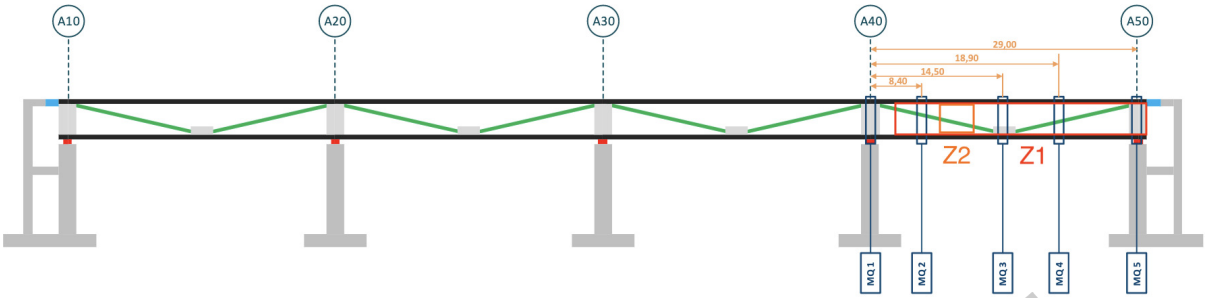

Fig. 4. Side view of the measurement cross sections [32].

Table 1. Overview of different damage scenarios in the Simulated Smart Bridge Dataset [32].

Scenario	Duration (2018)	Description
N	01.01, 00:00 – 08.07, 23:59	Healthy state
S1	09.07, 00:00 – 06.08, 23:59	Gradual reduction of stiffness of 30 % in Z1
S2	09.07, 00:00 – 06.08, 23:59	Gradual reduction of stiffness of 50 % in Z2
S3	09.07, 00:00 – 06.08, 23:59	Immediate increase in the bearing stiffness

Autobahn (digital highway test site) program under the auspices of the German Federal Ministry for Digital and Transport. The subject of observation in the smart bridge framework is a concrete highway bridge situated at a highway junction near Nuremberg, at the convergence of highway A3 from Frankfurt am Main to Regensburg and highway A9 from Ingolstadt to Bayreuth. The bridge spans four lanes, measuring a total length of 155 meters, and is supported by five concrete pillars, exhibiting a width of 15 meters.

During its construction in 2016, the eastern terminus of the bridge was outfitted with a wireless sensor network situated between the penultimate and final support columns. This network comprises 36 sensors strategically positioned across five distinct measurement cross-sections (see Figure 4), measuring accelerations, elongations, and displacements of various bridge components. Given the recent construction of the bridge and its early stage in the operational lifecycle, the recorded data is deemed representative of the undamaged, healthy state.

In practical terms, the lack of real-world damage data required resorting to Finite Element Method simulations to simulate the baseline behavior of the bridge. The term "baseline behavior" here refers to the fundamental performance or condition of the bridge under normal conditions. This modeled behavior serves as a reference or baseline for subsequent comparisons with actual observations or measurements, especially when it comes to identifying damages or anomalies and was calibrated step by step according to the real measurements. Additionally, simulations incorporating three distinct damage scenarios (N, S1, S2 and S3) were conducted in an additional simulation project [32]. An overview of these scenarios is presented in Table 1. This dataset consists of artificial measurement data generated through a numerical simulation of a bridge with simplified geometry, allowing for controlled experiments and the comparison of different deep learning models for structural health monitoring.

Just Accepted

Table 2. Available sensors in the Simulated Smart Bridge dataset [32].

Sensor label	Section	Location	Sensortype
MS_U_Neig	MQ1	below	Inclination Sensor
MS_U_Schieb	MQ1	below	Inclination Sensor
MS_U_MI_L_o	MQ2	top	Strain Gauge
Ms_U_Li_Int_u	MQ2	bottom left	Strain Gauge
MS_O_MI_L_u	MQ2	bottom	Strain gauge
MS_U_Neig	MQ2	below	Inclination Sensor
Ms_U_Re_Int_u	MQ2	bottom right	Strain Gauge
–	MQ3	Same sensor setup as MQ2	
–	MQ4	Same sensor setup as MQ2	
–	MQ5	Same sensor setup as MQ1	

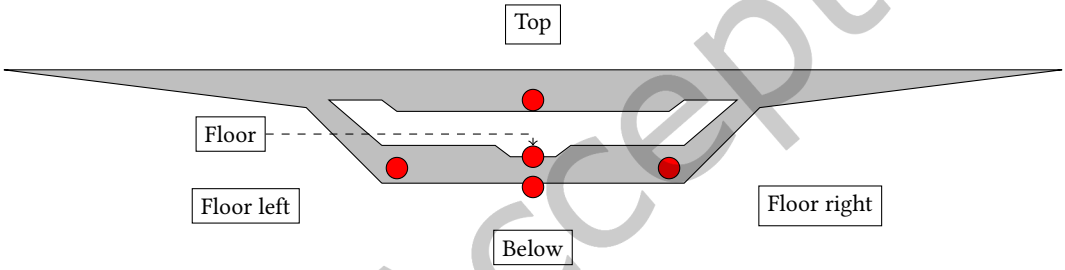

Fig. 5. Measurement cross-section of the simulated bridge structure [32].

The tabular overview of the sensortypes and locations is presented in Table 2. The strain gauges, strategically placed at different locations, measured the strain of the structure at a sampling rate of 75 Hz. Simulated environmental conditions included wind gusts modeled with a Gaussian stochastic process (mean wind speed of 30 m/s), traffic loads simulated through a moving force applied to the bridge, and temperatures ranging from -10°C to 40°C. The inclination sensors provide inclination values in milliradians.

The positions of the sensors within the cross sections are indicated in Figure 5. Specifically, the sensors have been placed either inside the hollow chamber on the floor at the center, on the floor to the right, on the floor to the left, below the concrete of the hollow chamber, or on top the concrete of the hollow chamber. Placing sensors at evenly distributed points across the hollow chamber ensures comprehensive coverage of the area under surveillance, minimizing the risk of overlooking important information.

3.2 Luxemburg Dataset

The Luxemburg dataset, as documented in Schommer et al. [69], comprises structural vibration measurements obtained from a decommissioned bridge that underwent various real-world damage scenarios. This bridge, originally constructed from 1953 to 1955, spanned the River Mosel, connecting the German city of Wellen to the

Just Accepted

city of Grevenmacher in Luxemburg.

The bridge consisted of five spans, each composed of five parallel concrete beams. Steel tendons interconnected neighboring beams, with additional transverse concrete beams placed every 7.5m along their longitudinal axis. Each beam was pre-stressed by tendons embedded into the concrete along its longitudinal axis.
The sensor network contained three different types of sensors: 26 accelerometers, eight displacement sensors, voltage sensors and temperature sensors. All displacement sensors were mounted to the bottom of the beam with their reference fixed to the ground below. Seven of these measured vertical displacement while SH7 measured horizontal displacement. The sensor network employed three types of sensors: 26 accelerometers, eight displacement sensors, voltage sensors, and temperature sensors. Displacement sensors were affixed to the bottom of the beam, with their reference fixed to the ground. Seven sensors measured vertical displacement, while one, denoted as SH7, measured horizontal displacement [69]. Temperature sensors were embedded into the concrete at a depth of at least 5cm, measuring the ambient temperature in the shadow. To simulate the original bridge's weight, weights totaling 30t were permanently placed on the left half of the beam.

To simulate the weight of the original bridge, several weights with a total mass of 30t were permanently placed on the left half of the beam.

The dataset encompasses both static and dynamic tests. Static testing involved temporarily placing up to four live loads of 13t each on the bridge. Dynamic tests assessed the vibration response under external excitation using a shaker (see Figure 6) , applying a swept sine wave with a sweep rate of 0.02 $^{Hz}/_{s}$ and a range from 2.5Hz to 10Hz with an excitation force of 2000N [69].

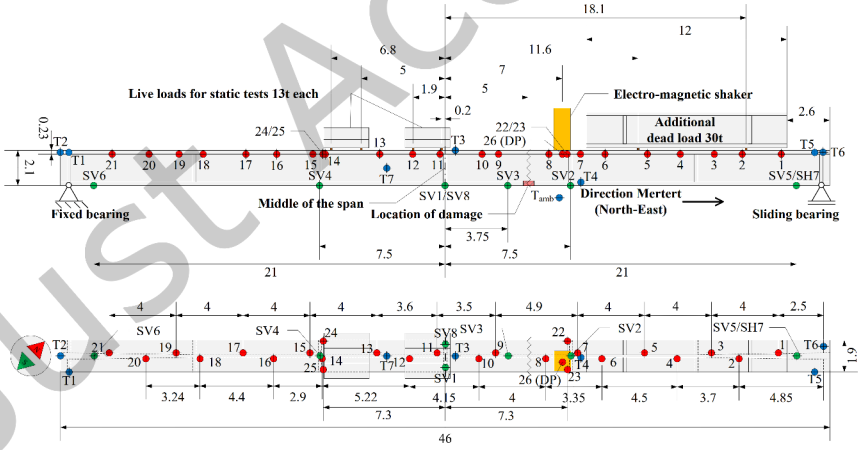

Fig. 6. Overview of the measurement setup in side view (upper view) and top viewer (lower part) with a, accelerometers in red; b, displacement sensors in green; c, temperature sensors in blue and d, shakers in yellow, [69].

The averaging of temperature values from seven sensors is done to represent an overall mean temperature for the structure. This approach is grounded in the principle of spatial averaging, where multiple measurements are combined to obtain an average or representative value. By utilizing several sensors, local variations or

Just Accepted

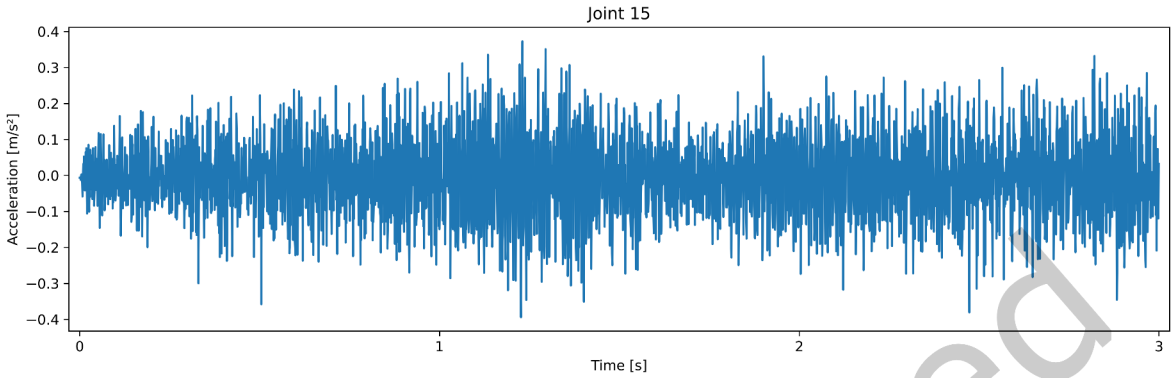

Fig. 7. Three-second sample of acceleration measurement shows high signal to noise ratio (SNR)

.

irregularities in the temperature distribution can be mitigated, and the averaged temperature provides a more stable and reliable representation of the overall structural temperature. This is particularly crucial when the structure is subject to varying environmental influences in different areas.

Acceleration data (see Figure 7), recorded at a sampling rate of $2500Hz$, is parsed from TDMS-files in the `DynamicMeasurements` folder using the npTDMS Python library[3].These measurements show to have a high signal to noise ratio (SNR) and are used as raw data input. Temperature measurements, occurring once per minute, are associated with the nearest acceleration measurement and feeded into the network as additional dynamic node features.

Following the bridge's demolition, two beams were transported to the port of Mertert in Luxemburg for experiments. The beams, each $46m$ in length and weighing approximately $120t$, underwent various damage scenarios involving cutting tendons and adding masses at specific locations. These scenarios, outlined in Table 3, encompass progressive damage stages. Sensor network details are illustrated in Figure 6.

3.3 Data Preprocessing

The total size of both datasets in their raw unzipped format amounts to 2.3TB for the simulated dataset and 68.7GB for the Luxemburg dataset. Besides the experiments conducted on these large datasets, that can be requested from the university of Luxembourg and from the German Federal Highway Research Institute (Bundesanstalt für Straßenwesen) in full size, we stored five percent of the datasets in csv format in the provided open repository[4]. We didn't exclude any data of the datasets.
In the case of the Luxemburg dataset, the signals have been recorded by a data acquisition system (DAQ) of National Instruments and stored in TDMS data format, while the smart bridge simulated data was delivered in a

[3]See https://nptdms.readthedocs.io/en/stable/
[4]https://github.com/MilanShao/ModeConv

Just Accepted

Table 3. Overview over the different damage scenarios in the Luxemburg Bridge Dataset [69]. The damage is applied progressively, i.e. the changes made in scenario S1 are still present in all following scenarios.

Scenario	Duration (2014)	Description
N	23.01, 00:00 – 31.01, 12:00	Healthy state
S1	31.01, 12:00 – 04.02, 12:00	2 tendons cut
S2	04.02, 12:00 – 06.02, 12:00	4 tendons cut
S3	06.02, 12:00 – 11.02, 12:00	6 tendons cut
S4	11.02, 12:00 – 19.02, 12:00	6 tendons cut, 6 tendons partly cut

binary data format. As the initial format of both datasets was not suitable for time-series analysis, we devised a preprocessing pipeline to transform the data into an intermediate format. For each day of measurements, we generate a separate file for each measurement channel of all sensors with the original sampling rates and resolutions. The data is stored, with each measurement encoded as its timestamp (4-byte integer) followed by the measured value at that time (4-byte float). Finally, we split the data into disjoint training, validation, and test sets as follows and normalize them. Over the course of 23 (non-consecutive) days and 52 experiments, data were collected for the normal behavior class (10 %) and the abnormal behavior class (90 %) containing all four damage classes. For training, experiments with IDs $[0, 1, 2, 3, 6]$ were used, while experiments with IDs $[5, 7, 22, 51]$ were allocated for validation. The remaining experiments were designated for testing. Experiments $[0 : 7]$ were characterized by normal behavior, while the rest were labeled as exhibiting at least one of the damage classes. In the case of the Simulated Smart Bridge dataset data were collected over a course of 217 days towards the classes N (189 days of normal behavior) and $S0$-$S3$ (28 days split between normal and three distinct damage scenarios representing abnormal behavior). This results in a distribution of approximately 80% normal behavior and 20% abnormal behavior. In the provided five percent dataset all classes have been used equally.

4 ModeConv Method

Vibration-based techniques are employed to assess the integrity of structures. We categorize vibrations into three types: free vibrations, which occur when a structure oscillates on its own, and forced vibrations, caused by external energy sources like wind gusts or traffic-induced vibrations [20, 71] as well as their overlapping vibrations (see Figure 8, Signal Block). By extracting eigenmodes, thus focusing on free vibrations of the structure, we can detect changes and damages in the structure [63, 68]. We therefore calculate the frequency response function of the PDE building block (see Figure 8, PDE building block) as well as the Cross Spectral Density (CSD) (see Figure 8, Signal Block) as input to the Convolutional Block (see Figure 8, Conv. Block). To enable anomaly detection in structures, ModeConv leverages statistical and spectral analysis techniques to analyze the vibration behavior. Specifically, it utilizes the modal transformation of the signals contained in the datasets to identify any deviations or anomalies in the structural response.

The method section is organized into the three aforementioned main building blocks of the overview figure: the PDE building block, the signal block, and the convolution block, that are depicted in the ModeConv Overview (see Figure 8). Each of these building blocks is explained in detail in the following subsections. Please note that whenever a calculation of one entry in a matrix is described, the notation is written without square brackets, while matrices for which every entry is calculated contain square brackets. All matrices are written with capital letters, while vectors are written with small letters, and tensors in bold letters.

Just Accepted

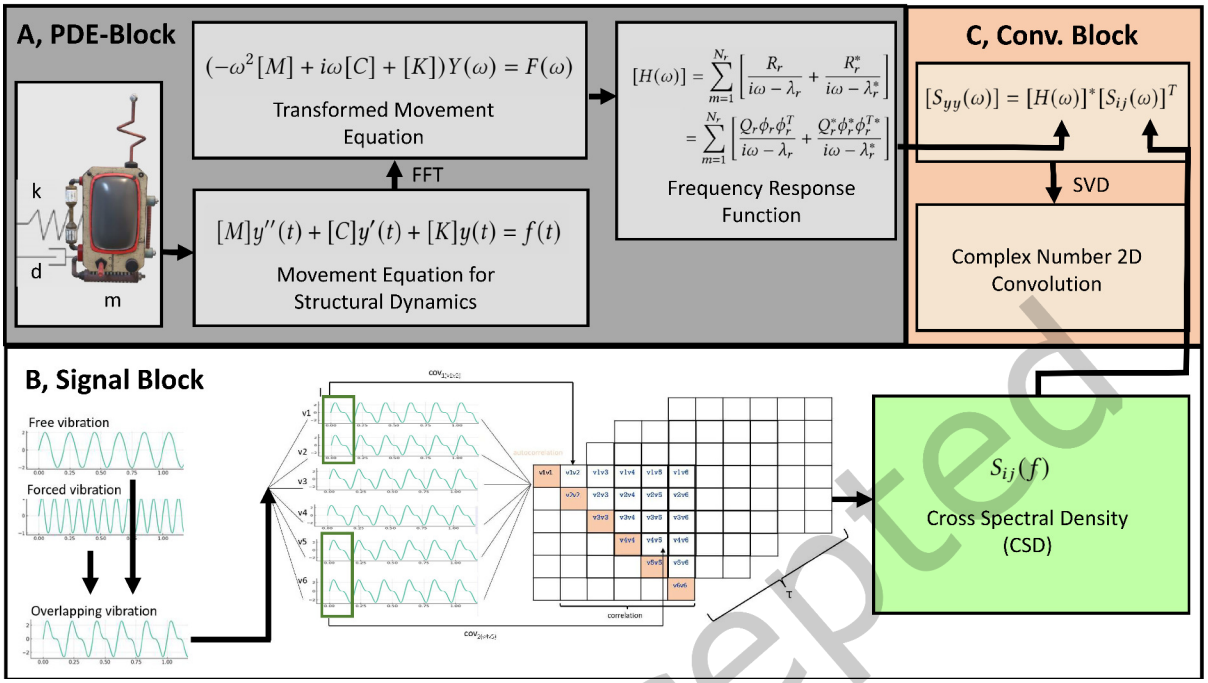

Fig. 8. Overview over ModeConv building blocks with its A, PDE Block, B, Signal Block and C, Convolutional Block.

Liu et al. highlighted the superiority of non-linear dimensionality reduction techniques, such as stacked autoencoders, over linear methods in extracting features from vibration data for damage diagnosis [53]. We instead utilize ModeConv in a Graph Autoencoder architecture for its adaptability to noise and structural variation without manual adjustments. Neural networks efficiently model complex interdependencies and non-linear dynamics, making them suitable for tasks that challenge traditional approaches [44]. Although neural networks can be computationally demanding, ModeConv reduces costs through efficient data compression and parallel usage of different threads on GPUs, outperforming traditional FEM in computation speed.

ModeConv mitigates the need for precise input assumptions in structural health monitoring by employing statistical mixtures of neural networks [38]. While techniques like stochastic subspace identification (SSI) are specialized, ModeConv excels in data-rich environments by detecting patterns and anomalies without extensive modeling assumptions. It also effectively learns correlations in complex datasets even with uncertain input parameters [38].

Just Accepted

4.1 Data Input

In applications of sensor networks, data naturally reside on the vertices and edges of weighted graphs (see Figure 8, in which the sensors represent the vertices and the connection between the sensor nodes can be represented by edges. This graph structure can be used in structure aware models such as graph convolutions.

A Graph is represented as $G = (V, E)$ where V is the set of vertices (v_i) representing sensor nodes and E is the set of edges between sensor nodes. The data on these graphs is represented as a finite set of samples, with each sample, known as a sensor measurement $x_i(t)$ for a sensor node v_i, associated with each vertex in the graph. The

edge index is denoted as $e = (v_i, v_j)$ where $0 \leq i, j < |v|$

In the ModeConv approach for determining the edge weights of the graph, we initially construct a fully connected graph, where each sensor node $v_i \in V$ is connected to every other sensor node $v_j \in V$ via an edge $e = (v_i, v_j)$. To ascertain the correlation between these nodes, we compute the covariance matrix of the sensor measurements $x_i(t)$. The covariance matrix provides a measure of how the sensor measurements co-vary with each other over time, reflecting the interdependencies among the sensor nodes.

For each pair of nodes (v_i, v_j), we derive a similarity score based on the corresponding entry in the covariance matrix. This score quantifies the strength of the correlation between the measurements at nodes v_i and v_j. We establish a threshold value θ to determine the presence of an edge: if the similarity score exceeds θ, an edge is maintained between v_i and v_j, otherwise, the edge is removed. This thresholding process allows us to convert the fully connected graph into a sparser graph, retaining only the most significant connections.

This adaptive method of updating the graph structure is iteratively performed, ensuring that in each iteration, the graph reflects the most current and relevant interactions between the sensor nodes based on the latest data observations.

Additional dataset-specific sensor readings can be introduced as vertex features in this graph model, depending on the dataset. In the Luxemburg dataset, it includes measurements of acceleration, temperature, voltage, and displacement. For voltage, temperature, and strain features, statistics such as mean, standard deviation, minimum, and maximum values are computed. In the smart bridge simulated damage dataset, we obtain strain-gauge, temperature, and acceleration measurements. One sample is obtained by concatenating the mean values of strain, temperature, and acceleration along the last dimension. The batch-size has been optimized towards 256.

Additionally, acceleration measurements are specifically used for pairwise signal covariance calculation, denoted as C_{ij} (illustrated as signals in Figure 8). Considering the number of sensors N in the network, a single sample is constructed by concatenating the mean values of voltage, displacement, and temperature along the last dimension.

4.2 Signal Block

In the context of signal processing, a crucial component lies within the signal block, where specific strategies are employed to process and analyze the data streams effectively. Within this signal block only the first sensor channel is used for the calculations, because these are the relevant data to calculate the eigenmodes while additional data like temperature show dependecies and are therefore just used as additional features. In the case of the Luxemburg dataset that is the acceleration measurement and in the case of the simulated dataset the strain gauges are used (see section above).

To ensure that the time series data can be evenly divided into overlapping windows of size l, the data is first padded. The hyperparameter l has been optimized for five samples, which corresponds to an aggregation interval of microseconds given a sampling rate of 2500 samples per second. Larger aggregation steps were found to result in decreased performance, despite the desire to reduce the size of the large datasets. Therefore l was kept small. Then the covariance C_{ij} for pairwise signals is computed for all signals x_i and x_j, where i and j is used to seperate the measurements of the sensors i and j, over the length of l samples. In this equation, x_i and x_j are two different signals i and j and $x_i(t)$ is the value of the signal x_i at time-step t for a sensor node v_i.

$$X_i = \{x_i(l) : 0 \leq l < 5\}$$

Just Accepted

$$X_j(\tau) = \{x_j(l+\tau) : 0 \leq l < 5\}$$
$$C_{ij}(\tau) = E[(X_i - E[X_i]) - (X_j(\tau) - E[X_j(\tau)])]$$
$$C_{ij}(\tau) = E[(x_i(l) - \mu_i)(x_j(l+\tau) - \mu_j)] \quad (1)$$

The transformation from the cross-correlation $C_{ij}(\tau)$ to the cross-correlation $R_{ij}(\tau)$ is achieved through scaling. The cross-correlation $R_{ij}(\tau)$ is defined as the normalized version of the covariance function $C_{ij}(\tau)$.

To transition from $C_{ij}(\tau)$ to $R_{ij}(\tau)$, we divide $C_{ij}(\tau)$ by the square root of the product of the variances of signals x_i and x_j [55]:

$$R_{ij}(\tau) = \frac{C_{ij}(\tau)}{\sqrt{C_{ii}(0)C_{jj}(0)}} \quad (2)$$

Here, $C_{ii}(0)$ represents the autocovariance of signal x_i, and $C_{jj}(0)$ represents the autocovariance of signal x_j at zero time lag ($\tau = 0$). This normalization yields the normalized cross-correlation function $R_{ij}(\tau)$, which is commonly used for further analysis.

The Fourier transformation of the cross-correlation $R_{ij}(\tau)$ yields the cross spectral density (CSD) $S_{ij}(f)$, describing the distribution of power across frequencies in the frequency domain.

$$S_{ij}(f) = \int_{-\infty}^{\infty} R_{ij}(\tau) e^{-i2\pi f\tau} \, d\tau \quad (3)$$

This relation is called the Wiener-Chintschin relation, which is a mathematical relationship that allows to analyze the cross spectral density (CSD) of structural response signals. This provides valuable information about the distribution of power across different vibration frequencies[41].

Calculating covariances, cross-correlations, and cross spectral density (CSD) in the Signal Block provides deeper insights into the relationships and structures within the sensor data. This allows to recognize patterns, identify trends, and gain insights into the behavior of the system. CSD offers information about the distribution of power across various frequencies. Covariance and cross-correlation reveal the relationships between different signals or sensors. Analyzing cross-correlation enables the detection of changes in system behavior over time, facilitating early detection of anomalies indicating potential problems or deviations in the system.

4.3 PDE Building Block

The equation of motion describes the dynamics of a system and enables the determination of its natural frequencies, damping ratios, and modes. By modeling the equation of motion of a system, it is possible to develop a mathematical understanding of the system behavior. This is particularly useful for analyzing the structural or material properties of constructions or machinery, such as bridges, buildings, aircraft, or turbines.

The PDE building block (see Figure 8) makes use of this motivation and expresses the dynamic behavior of a system in the time domain using the equation of motion [27]. In the figure, d denotes damping, m denotes mass and k denotes stiffness. The equation of motion for structures includes a vector of excitation forces, denoted as $f(t)$. In both datasets, the mass of the structure or infrastructure is divided into points of mass according to the number of sensors, and this sums up in a mass matrix $[M]$. The mass matrix $[M]$ is used for analyzing the behavior of structures subjected to dynamic loads such as earthquakes, wind, or traffic. It is a diagonal matrix with the mass of the structure, proportionally allocated to each sensor mass point denoted by m as one mass point on the diagonal, i.e. the degrees of freedom are equal to the number of sensor nodes. In this context, each node $|v|$ of the graph possesses its individual mass denoted by m. $y''(t)$ denotes the vector of acceleration, $y'(t)$

Just Accepted

the vector of velocity and $y(t)$ the vector of displacement. The dimensions of the stiffness matrix $[K]$ of the structure are equal to the number of sensors in the network, and as a bridge can be defined as a one-mass-swinger it follows a series connection. In this context, a series connection refers to the configuration where the bridge is considered as a sequence of interconnected elements. Here, the number of sensors in the network corresponds to the dimensions of the stiffness matrix $[K]$, implying that each sensor serves as a component in this series arrangement. The damping of the system is expressed through the damping matrix of the structure $[C]$:

$$[M]y''(t) + [C]y'(t) + [K]y(t) = f(t) \tag{4}$$

The stiffness matrix $[K]$ is defined in series connection as [45]:

$$[K]_{(i,j)} = \begin{cases} 2k & \text{if } i = j \\ -k & \text{if } |i - j| = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The dimensions of the stiffness matrix $[K]$ are also equal to the number of sensors in the network.

In order to decouple the equation of motion Equation (4), it is transformed to the modal form [62]. The basic idea is to express every oscillation state of a system by a combination of its natural modes. This is achieved by transforming the time-dependent displacements into a modal coordinate frame [27]

$$x(t) = [\phi]p(t) \tag{6}$$

where $[\phi]$ denotes the modal matrix and $p(t)$ the modal coordinates.
Then the eigenvalue problem is solved [63]. This returns the natural frequencies f_n and mode shapes ϕ. The mode shapes are then normalized, so that the maximum value of each column is equal to 1. Next, the mode shapes are ordered by ascending natural frequency.
The modal mass matrix $[M_{modal}]$ is calculated by projecting the mass matrix onto the normalized mode shapes[27]. The same is done for the modal stiffness matrix $[K_{modal}]$

$$[M_{modal}] = [\phi]^T \Lambda [\phi] \tag{7}$$

$$[K_{modal}] = [\phi]^T \Gamma [\phi], \tag{8}$$

Here, the damping ratio ξ is set to 0.02 for concrete material. The modal damping matrix $[C_{modal}]$ is obtained by transforming $[C_{modal}]$ into the coordinate system of the original mass matrix using the inverse of the mode shape matrix $[\phi]^{-1}$.

The mode shape matrix $[\phi]$ represents the spatial patterns of vibrations associated with the natural frequencies of the system. By utilizing these mode shapes, the modal mass, stiffness, and damping can be expressed in a coordinate-system-friendly framework, facilitating the analysis and interpretation of structural properties.

To diagonalize the damping matrix $[C]$ with the eigenvectors of the undamped system, the convenience hypothesis [22] must be introduced. In this case, proportional damping is assumed. In the special case of Rayleigh damping [54], $[C]$ can be expressed as a linear combination of the mass and stiffness matrices.

Then the equation of motion in frequency space can be denoted as:

$$(-\omega^2[M] + i\omega[C] + [K])Y(\omega) = F(\omega) \tag{9}$$

Just Accepted

Here, $Y(\omega)$ represents the response of the system in frequency domain, while $F(\omega)$ represents the external excitation force. $[M]$, $[C]$, and $[K]$ are the mass, damping, and stiffness matrices of the system. We follow the calculation of Rainieri et al. [65]

The matrix of system responses $[H(\omega)]$ is expressed as a sum of resonance responses. Using partial fraction decomposition $[H(\omega)]$ can be expressed in its modal form [65]:

$$[H(\omega)] = \frac{Y(\omega)}{F(\omega)} = \frac{\text{adj}(-\omega^2[M] + i\omega[C] + [K])}{\det(-\omega^2[M] + i\omega[C] + [K])} \tag{10}$$

The matrix of system responses $[H(\omega)]$ can be denoted as [65]:

$$[H(\omega)] = \sum_{r=1}^{N_r} \frac{[R_r]}{i\omega - \lambda_r} + \frac{[R_r^*]}{i\omega - \lambda_r^*} = \sum_{r=1}^{N_r} \frac{Q_r \phi_r \phi_r^T}{i\omega - \lambda_r} + \frac{Q_r^* \phi_r^* \phi_r^{T*}}{i\omega - \lambda_r^*} \tag{11}$$

Here N_r represents the number of mode shapes of the system, while $\lambda_r = \sigma_r + i\omega_{d,r}$ is denoted as the pole of the r-th mode and ϕ_r is the mode shape of the r-th eigenmode, while Q_r denotes the scaling factor. The scaling factor Q_r is introduced to modulate the contribution of the r-th mode to the overall system response. Specifically, Q_r is multiplied with the outer product of the mode shape ϕ_r in the expression for the matrix of system responses $[H(\omega)]$. The role of Q_r is to control the amplitude or intensity of the r-th mode within the total system response. Its introduction allows for the adjustment of the influence of each mode on the dynamic behavior of the system. Please note that the $*$ is used for the complex conjugate, λ_r represents the conjugate pole element of the r-th mode, ϕ_r denotes the conjugate mode shape element of the r-th eigenmode, and Q_r^* signifies the complex conjugate scaling factor of the r-th mode. R_r represents the contribution of the $r - th$ mode in the frequency domain. It denotes the residue term of the partial fraction decomposition of the frequency response function.

4.4 Convolution Block

At the core of ModeConv's architecture is a convolutional filter uniquely designed to process complex signals by incorporating both the frequency response function (FRF) and CSD. By doing so, the neural network can attenuate noise and amplify the distinct eigenmodes inherent to the structure, facilitating the differentiation between normal and anomalous behaviors. In the Convolution Block, the Cross Spectral Density (CSD) matrix $[S_{ij}]$ in its diagonal form is obtained from the Signal Block (see Figure 8). It is element-wise multiplied with the complex-valued weighting matrix $[H(\omega)]$ to obtain the weighted CSD matrix $[S_{yy}] \in \mathbb{C}$. It is computed as follows [65]:

$$[S_{yy}(\omega)] = [S_{ij}(\omega)][H(\omega)]^T \tag{12}$$

Here, S_{ij} denotes the cross spectral density matrix as given in Equation (3) [65]. In the literature, there can be found different versions of multiplications for these matrices [55, 65]. It is important to note, that the cross spectral density matrix is typically symmetric. Therefore, transposing $[S_{ij}(\omega)]$ yields the same information than $[S_{ij}(\omega)]$. Therefore, this results in the weighting matrix when multiplied with $[H(\omega)]^T$.

The multiplication of the Cross-Spectral Density (CSD) by the Frequency Response Function (FRF) in ModeConv aims to enhance resonant frequencies by using the FRF as a filter. This process emphasizes structural modes that correspond to natural resonances (where FRF values are highest), making them more prominent in the CSD. Simultaneously, it attenuates frequencies not aligned with the structural characteristics due to their lower FRF magnitudes. The modulation results in an optimized input that highlights informative frequencies, enhancing the subsequent neural network layers' ability to process relevant data for anomaly detection. While typically the

Just Accepted

output spectral density is divided by FRF for back-calculating input characteristics, the multiplication approach taken here utilizes the FRF to filter out background noise, emphasizing the significant modes of vibrational response. This aids in aligning the analysis closer to the operational modal parameters, which are more relevant in fault detection scenarios.

4.5 Convolutional Filter

In Graph Signal Processing, the goal of the convolutional filter is to capture the underlying structure or patterns within the graph data. When dealing with modal coordinates, that are uncorrelated, $[S_{yy}(\omega)]$ shows a diagonal structure, wherein all values outside of the diagonal are zero. We then perform a singular value decomposition (SVD) on $[S_{yy}(\omega)]$ for specific frequencies per batch. This allows $[S_{yy}(\omega)]$ to be factorized and can be expressed as:

$$[S_{yy}(\omega)] = [U][\varepsilon][V]^T \tag{13}$$

Here, diag$[\varepsilon]$ contains the singular values in descending order, while $[U]$ and $[V]$ contain the singular vectors. Direct comparison with modal decomposition shows that the individual singular values of the SVD can be regarded as a measure of the contribution of each mode shape to the overall system response. The singular vectors represent the mode shapes of the system. Therefore, the system can be decomposed into as many single-degree-of-freedom oscillators as the system has degrees of freedom, and their interaction can be expressed through the modal coordinates in the form of $[\varepsilon]$.

Following singular value decomposition, the matrix $[U]$ is used to compute the convolutional filter weights, reflecting each eigenmode's contribution to the data's structure and dynamics. These weights feed into the ModeConv layer of a graph neural network. Then, the average over the second dimension of u is computed to obtain the weight matrix.

Weights W are complex, comprising real (W_r) and imaginary (W_i) components, used in the message-passing process to accurately convey significance between nodes. ModeConv Fast utilizes the SVD-derived eigenmodes tensor and the input tensor by reshaping and performing a tensor product, optimizing for a number of n eigenmodes.

For message passing, weights W are element-wise multiplied by similarity values, representing node correlations. The input x is multiplied by the computed weights:

$$x = X \cdot \text{weight} \tag{14}$$

$$y = (W_r \cdot x_r) - (W_i \cdot x_i) + b \tag{15}$$

Here, y combines real and imaginary contributions of the weights. The final model output aggregates y from each layer, allowing the model to learn complex patterns for tasks like anomaly detection on graph data, thus leveraging domain-specific knowledge. Separating real and imaginary components enables complex number convolution, enhancing pattern-capturing capabilities.

4.6 Time delay based edge weight update

As shown in Figure 9, ModeConv adapts edge weights dynamically based on the covariance of sensor data, offering a novel way of learning the physical relationships between sensor nodes. This method captures the inherent structural dynamics more accurately than fully connected graph models.

As weighted graphs are irregular structures that lack a shift-invariant notion of translation, we use the covariance of the sensor network from the signal block to correlate the signals, i.e., to determine to what extent two signals x_i and x_j depend on each other (see Signal Block). Thus, the autocorrelation and correlation of the signals are used to initialize and update the edge weights. The averages of the covariance across the sensor values

Just Accepted

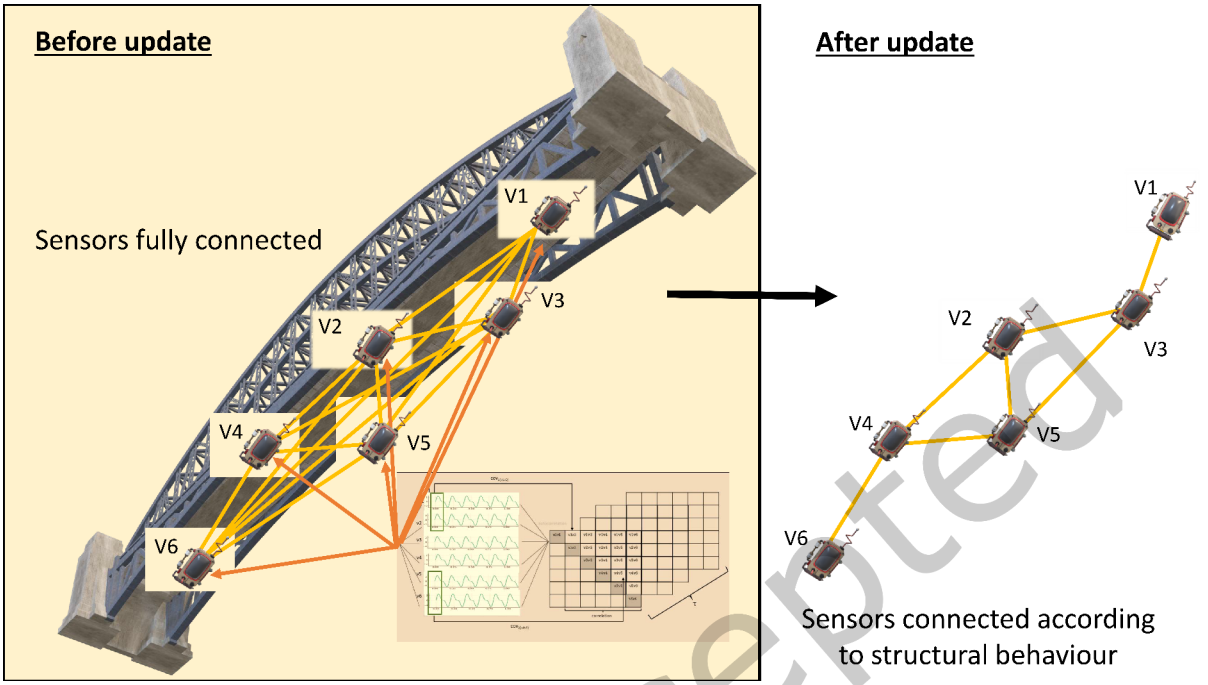

Fig. 9. Overview over ModeConv Edge Weights: the correlation and autocorrelation of signals is taken to update the edge weights. While it starts with a fully connected graph (see left side), the updated edge-connections follow the structural behvaiour (see right side).

Just Accepted

are taken for each instance in the batch to calculate the similarity. According to the similarity of the sensor values, irrelevant edge indices are filtered out and the nodes are not connected in the graph, while relevant nodes according to the inherent structure are linked. In ModeConv, covariance is efficiently used for both computing the cross spectral density (CSD) and constructing graph edge weights, offering dual utility that transforms the data and shapes the graph structure. This is an advantage over methods like STFT or wavelet transforms, which while effective for frequency analysis, cannot directly be used for graph connectivity based on signal dependencies.

4.7 Anomaly detection with ModeConv layers

Our proposed ModeConv layer is designed to be flexible and can be integrated into a variety of architectures. Here, in the case of anomaly detection, all benchmarking models are conditioned on an (semi-)supervised surrogate task such as reconstructing a given output through a bottleneck or predicting the evolution of an input in the future. During inference, anomalies can then be detected via a threshold mechanism on the reconstruction/prediction errors or residuals of the model. Given a multi-variate time-series with signals $x_{ij} \in \mathbb{R}$, the model M is asked to predict the multi-variate time-series $Y \in \mathbb{R}$ immediately following x_{ij}. Its prediction $\hat{Y}$ is then compared to the ground-truth Y via a loss function L. Here, the MSE between Y and $\hat{Y}$ are used.

In this work, anomaly detection is conducted within a semi-supervised learning framework. A graph autoencoder (GAE) is employed, in which both the encoder and decoder are composed of ModeConv layers. The GAE with ModeConv layers is then trained exclusively on data representing normal operating conditions. During

training, it learns to reconstruct input features by encoding and decoding through the ModeConv layers, thereby internalizing the underlying normal patterns.

After training, the reconstruction error is computed for each instance in the test set. Since the model has not been exposed to anomalous patterns, such instances are expected to yield significantly higher reconstruction errors.

To classify an instance as anomalous, a threshold is determined based on the distribution of reconstruction errors on a validation set. Specifically, the 95th percentile of this distribution is used as the anomaly threshold. This method ensures that the model remains sensitive to deviations from normal behavior, while keeping the false-positive rate low. A higher reconstruction error indicates a greater deviation from normal patterns. Once the model is trained, thresholds are set to determine at what point a reconstruction error is considered an anomaly.

Here, the effectiveness of two different methods for setting the threshold in anomaly detection tasks is assessed: the L1 threshold [57] and the Mahalanobis distance [23] calculated at the 95th percentile. These measures are employed to detect anomalies in various models. The L1 threshold method involves determining a threshold value based on the L1 norm. The L1 norm represents the sum of the absolute values of the elements in a vector. In this context, the L1 norm is calculated for each data point, and a threshold is set. Any data point with an L1 norm exceeding this threshold is considered an anomaly. The Mahalanobis distance is calculated at the 95th percentile. This distance provides a measure of how far a data point deviates from the mean in multivariate space, considering the covariance structure of the data. Points beyond the 95th percentile threshold are identified as anomalies. The Mahalanobis distance is particularly suitable for situations with a larger variance within the data, as it accounts for correlations between different features. In the anomaly detection process, each data point is evaluated based on the chosen method (L1 threshold or Mahalanobis distance). If the calculated metric for a data point exceeds the set threshold, the point is flagged as an anomaly. This process enables the identification of unusual patterns or outliers within the dataset.

On the other hand, the Mahalanobis distance is a measure of the distance between a data point and the mean of a distribution, taking into account the covariance matrix. We calculate the Mahalanobis distance for each data point and compare it against the Mahalanobis distance at the 95th percentile. If the Mahalanobis distance of a data point exceeds the 95th percentile value, it is identified as an anomaly.

5 Experiments

We propose two major advantages of ModeConv, which need to be validated in the experimental section. First, ModeConv delivers better results on anomaly detection tasks that rely on physical material properties and their monitoring compared to existing graph neural network models for multivariate time-series. Second, ModeConv reduces the data to the modal coordinates and uses SVD instead of the normalized Laplacian while retaining relevant information, resulting in better performance in terms of runtime.

To validate these claims, we compare the results of a graph autoencoder with ModeConv layers trained on the Luxembourg dataset with the results of the same graph autoencoder using ChebConv layers [26], which are a type of Spectral Graph Convolution. We also evaluate the results on the Simulated Smart Bridge dataset. In addition, we compare the performance of ModeConv autoencoder for the whole datasets with two state-of-the-art Graph Neural Networks for multivariate time-series, namely AGCRN [7] and MtGNN [80], both of which are built on Spectral Graph Convolutions [19]. For the five percent dataset we also compare it against the MTGODE model and the GraphCON wrapper.

Just Accepted

All of the models have been tuned via a hyperparameter study using optuna [3], that aims to improve the reconstruction error for all networks in order to create a fair experimental setup. The parameters included in the paper have been learning rate, chebyshev filter size K from 2, ..., 8, number of layers ranging from 1, 3, 5, 10, the number of hidden dimensions ranging from 4, 8, 16 and the bottleneck ranging from 1, 2, 4. Learning rate and dropout have been optimized according to the default values of Pytorch Lightning.

For the GraphCON model [67] the following formalization is used as foundation for the wrapper:

$$X'' = \sigma(F\Theta(X, t)) - \gamma X - \alpha X',$$

where σ is the activation function, $F\Theta$ is the 1-neighbourhood coupling function, and α, γ are learnable hyperparameters providing dampening. X denotes the time-dependent matrix of node features over time t, which serves as the main input for the forward function between GNN layers. In the case of GraphCON, we therefore also tune the learnable hyperparameters

$$0 \leq \alpha, \gamma \leq 2.$$

In their GitHub repository[5], Rusch et al. provide a code snippet, that had to be modified for the experiments to use it as wrapper for the MtGNN model.

The existing MtGNN architecture relies on the trainer functionality provided by PyTorch Lightning, which necessitates the model to be a PyTorch Lightning module. This requirement is not met by GraphCON. Consequently, we incorporated a wrapper directly into the message passing of MtGNN. However, GraphCON specifically expects graph convolutional layers with stable dimensions as input.

The MtGNN model consists of alternating graph convolutions (GC) and temporal convolutions (TC). The TC includes Dilated Inception Layers that lead to dimension reduction. This posed a challenge as GraphCON requires stable dimensions for its input. To address this, we made additional adaptations. In order to ensure that the dimensions of X and Y remain equal across iterations, a TC module is added for Y.

Moreover, we compare our ModeConv Fast version without normalized Laplacian with the ModeConv Laplace version in all experiments. Every model has been trained for 50 epochs. All experiments have been conducted on an NVIDIA GeForce GTX 1080 Ti GPU.

Furthermore, we compare the training time of all the models and conduct a comparative study on the single blocks of the ModeConv approach. This study evaluates the time-consuming and cost-improving aspects of the layer to accurately measure the complexity and performance improvement.

6 Results

To compare the results, several standard performance measures including precision, recall, F1 score, balanced accuracy, AUC (Area Under the Curve), and training time per epoch have been used. Precision indicates the accuracy of anomaly detection, with higher values representing fewer false positives while recall measures the ability to detect actual anomalies, with higher values indicating better performance. F1 Score is a balanced measure of precision and recall, with higher scores indicating a better trade-off between the two. Furthermore we used balanced accuracy, that calculates the average accuracy for both positive and negative classes, considering both highly imbalanced datasets used in these experiments. Moreover, AUC (Area Under the Curve) represents

[5]https://github.com/tk-rusch/GraphCON

Just Accepted

the model's ability to distinguish between positive and negative samples [46]. The time taken for each epoch during training, measured in hours has been tracked to document the energy costs.

6.1 Results Comparative Study

An comparative study was conducted on the Luxemburg Dataset to analyze the ModeConv layer's components and their impact on performance. Two variations were examined: replacing Cross Spectral Density (CSD) calculation with two linear layers (LL) in the signal block, and using normalized Laplacian instead of Single Value Decomposition (SVD) of the Convolutional block for the complex filter based on Laplacian weights.

Table 4. Performance metrics for different variants of the ModeConv model performance on the Luxemburg dataset. The best results are highlighted in grey.

Model Variants	AUC	Bal. Acc.	F1	Prec	Recall
ModeConv with SVD and CSD	0.91	0.87	0.90	0.98	0.84
ModeConv with Laplacian	0.91	0.87	0.90	0.98	0.84
ModeConv with LL and Lap.	**0.92**	**0.88**	**0.91**	**0.99**	**0.85**

The performance of the different ModeConv variants was evaluated, and the results are summarized in Table Table 4. All variants achieved similar performance. While the comparative study shows comparable performance, the speed or computational efficiency of the ModeConv with SVD and CSD variant, which is used as ModeConvFast in the following experiments, is higher whenever K is low (see following tables). This evaluation shows, that the modal transformation delivers comparable results to the spectral transformation.

6.2 Results Luxemburg Dataset

Regarding the whole Luxemburg dataset, the results for different models using the 95th percentile of L1 distance as the threshold are presented in Table Table 5.
ModeConvFast outperformed other models, achieving the highest F1 score of 0.949, perfect precision, and a

Table 5. Performance metrics for different models on the Luxemburg dataset with L1 distance. Best results highlighted in grey.

Model	Prec.	Recall	F1	Bal. Acc	AUC	Time (h/epoch)
ModeConvFast	1.0	0.903	0.949	**0.952**	**0.985**	**2**
ModeConvLaplace	0.99	**0.904**	**0.949**	**0.952**	0.984	19
ChebConv [25]	1.0	0.902	**0.950**	0.948	0.983	19
MtGNN [82]	1.0	0.900	0.947	0.950	0.976	15
AGCRN [8]	1.0	0.898	0.946	0.949	0.973	18

recall of 0.903. It also obtained the highest balanced accuracy of 0.952 and AUC of 0.985, with the lowest training time per epoch, while the training time per epoch with 2 hours versus the highest value between 15-19 hours was 7-9 times higher. As the Laplacian normalization is used in ChebConv as well as ModeConvLaplace the training times show off to be similar.

Furthermore, when using the Mahalanobis distance as the threshold, ModeConvFast achieved an F1 score of 0.939 and a recall score of 0.907, making it the best-performing model. ModeConvLaplace obtained the highest

Just Accepted

precision score of 0.994 and a balanced accuracy of 0.868, while ChebConv and AGCRN also showed competitive performance.

Table 6. Performance metrics for different models on the Luxemburg dataset with Mahalanobis distance. Best results highlighted in grey.

Model	Prec.	Recall	F1	Bal. Acc	AUC
ModeConvFast	0.974	**0.907**	**0.939**	0.665	0.911
ModeConvLaplace	0.994	0.865	0.925	**0.868**	**0.935**
ChebConv [25]	0.993	0.554	0.711	0.735	0.765
MtGNN [82]	0.500	-	0	-	0.500
AGCRN [8]	**1.0**	0.543	0.704	0.772	0.771

In summary, the choice of threshold metric has a notable impact on model performance. ModeConvFast demonstrated superior performance with L1 distance, while ModeConvLaplace excelled with Mahalanobis distance. The selection should be based on specific requirements, considering precision, recall, balanced accuracy, and training time trade-offs (Table 6).

6.3 Results Simulated Smart Bridge Dataset

Regarding the whole Simulated Smart Bridge Dataset when using the L1 distance to choose the threshold, the following results are gained:

Table 7. Performance metrics for different models on the Simulated Smart Bridge dataset with L1 distance. Best results highlighted in grey.

Model	Prec.	Recall	F1	Bal. Acc	AUC	Time (h/epoch)
ModeConvFast	**0.688**	0.618	**0.626**	0.754	0.826	**1**
ModeConvLaplace	0.531	**0.661**	0.603	**0.757**	**0.829**	15
ChebConv [25]	0.271	0.615	0.376	0.601	0.636	14
MtGNN [82]	0.630	0.603	0.616	**0.757**	0.828	16
AGCRN [8]	0.403	0.514	0.452	0.662	0.724	8

The ModeConvLaplace model achieved the highest balanced accuracy score of 0.757, indicating that it performed well on both the positive and negative classes. Its AUC score of 0.829 was also the highest among the models. The ModeConvFast model achieved the highest precision score of 0.688, which indicates that it had a low false positive rate. However, its recall score of 0.618 was lower than that of the ModeConvLaplace model, which achieved the highest recall score of 0.661. The ModeConvFast model also had the highest F1 score of 0.626, indicating that it had a good balance between precision and recall. The AUC score of the ModeConvFast model was 0.826, which was the second highest among the models. Again the training time per epoch is much faster than the runtime of all other models (see Table 7).

When using the Mahalanobis distance instead we gain the following results:

Among these models, ModeConvFast and ModeConvLaplace stand out with precision scores of 0.997, showcasing their ability to accurately identify anomalies. ModeConvFast achieves an impressive F1 score of 0.977, while

Just Accepted

Table 8. Performance metrics for different models on the Simulated Smart Bridge dataset with Mahalanobis distance. Best results highlighted in grey.

Model	Prec.	Recall	F1	Bal. Acc	AUC
ModeConvFast	**0.997**	0.923	**0.977**	0.949	0.978
ModeConvLaplace	**0.997**	0.958	0.962	0.960	0.975
ChebConv [25]	-	0	-	0.5	0.5
MtGNN [82]	0.895	**0.968**	0.930	**0.970**	**0.996**
AGCRN [8]	0.792	0.933	0.857	0.936	0.985

ModeConvLaplace excels in recall with a value of 0.958, highlighting its capability to capture a high proportion of actual anomalies. MtGNN also demonstrates strong performance with a balanced accuracy of 0.970 and an AUC of 0.996 (see Table 8).

6.4 Results Luxemburg Dataset 5%

For the experiments with the reduced dataset, we additionally compare the results against the MtGNN with GraphCON wrapper and against MTGODE in terms of F1 Score:

Table 9. F1 Scores for Different Models

Model	F1
ModeConvFast	**93.09**
ModeConvLaplace	73.74
ChebConv [25]	74.53
AGCRN [8]	86.67
MtGNN [82]	82.00
MtGNN [82] with GraphCON [67]	92.20
MTGODE [50]	66.00

ModeConvFast achieves the highest F1 score of 93.09, outperforming other models. ModeConvLaplace, ChebConv, and AGCRN exhibit F1 scores ranging from 73.74 to 86.67. MtGNN achieves an F1 score of 82.00, whereas the incorporation of the GraphCON wrapper improves its performance to 92.20. In contrast, MTGODE lags behind with an F1 score of 66.00. In comparison to the outstanding results of the MTGODE paper, the model performs less competitively on this dataset. One potential reason for this could be the presence of outliers and noise among the data samples, which were not included in the datasets used in the original MTGODE paper. It is likely that the training datasets in the original MTGODE paper did not contain as many outliers and noise as the datasets used in the Luxemburg and simulated smart bridge datasets (see Table 9).
What we also observe is a notable enhancement in the performance of ModeConv when computing on the reduced dataset compared to other models. This improvement accentuates its robustness, suggesting that the model excels in handling both small and large datasets. This versatility positions ModeConv as choice for anomaly detection tasks, showcasing its ability to maintain efficacy even with limited data.

Just Accepted

6.5 Results Simulated Smart Bridge Dataset 5%

As the complete dataset is several terabytes in size and thus computationally demanding to process, we also conducted experiments on 5 percent of the dataset. As well as for the reduced Luxemburg dataset, we also conducted experiments on the reduced simulated smart bridge dataset. Additionally, we integrated the GraphCON approach here, as it closely resembles ModeConv and therefore represented a relevant extension for our experiments.

Table 10. F1 Scores for different Models

Model	F1
ModeConvFast	87.93
ModeConvLaplace	88.07
ChebConv [25]	83.89
AGCRN [8]	87.76
MtGNN [82]	86.78
MtGNN [82] with GraphCON [67]	**90.67**
MTGODE [50]	63.00

The presented results underscore the efficacy of the GraphCON approach, which corresponds to the author's advice against simply stacking Graph Neural Networks (GNNs) [67]. The result validate the potential for significant enhancements in existing GNN architectures through GraphCON. While GraphCON introduces two additional learnable hyperparameters demanding careful optimization for optimal results, the computational load is the primary drawback. Although GraphCON's direct impact on training time is marginal, its true benefits emerge with increased model depth, amplifying the number of trainable parameters and subsequently extending training time. ModeConv Fast demonstrates 3 percent less effective results to the GraphCON wrapper while significantly reducing runtime (see Table 10).

Algorithm 1 GraphCON Forward Pass

procedure GraphCON($GNNs, X0, Y0$, edge_index, dt, α, γ, dropout)
  $X \leftarrow X0$
  $Y \leftarrow Y0$
  for gnn **in** $GNNs$ **do**
    $Y \leftarrow Y + dt \cdot (\text{ReLU}(gnn(X, \text{edge_index})) - \alpha \cdot Y - \gamma \cdot X)$
    $X \leftarrow X + dt \cdot Y$
  end for
  if dropout $\neq$ None **then**
    $Y \leftarrow \text{Dropout}(Y, \text{dropout})$
    $X \leftarrow \text{Dropout}(X, \text{dropout})$
  end if
  return X, Y
end procedure

The pseudocode in Algorithm 1 outlines the forward pass of the GraphCON wrapper, which is designed to capture dynamic behaviors through the application of Ordinary Differential Equations (ODEs). The algorithm

Just Accepted

utilizes a series of Graph Neural Network (GNN) layers to iteratively update two variables, denoted as X and Y. The update process is governed by an Implicit-Explicit (IMEX) scheme, providing a numerical solution to the ODEs. Furthermore, the algorithm supports optional dropout regularization during training. This pseudocode serves as a representation of the underlying PyTorch implementation, adapted for this specific setting.

7 Results of the additional benchmarking models

In this section different state of the art models for anomaly detection in time-series are used additionally and the benchmarking results are presented. The additional models contain MLP, VRAE, TCNAE, TGCN and STGCN. MtGNN, AGCRN and ModeConv are inserted in the chapters before.

The following table shows the results of ModeConv in comparison to these models for the Luxemburg dataset:

Table 11. Results for the complete Luxemburg dataset: ModeConv with L1 distance against other state of the art models

Model	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
MLP [34]	74.71	10.91	30.74	68.30
VAE [51]	56.32	10.89	26.72	65.24
TCNAE [75]	74.86	18.72	25.76	63.86
TGCN [88]	77.24	1.09	21.32	55.29
STGCN [85]	38.89	3.64	16.73	55.71
MTGNN [80]	59.29	6.63	33.73	67.69
AGCRN [8]	54.84	8.67	34.38	61.82
ModeConvFast	**99.99**	**90.30**	**94.90**	**95.20**

In analyzing further results on the Luxemburg dataset, ModeConvFast emerged as the top-performing model (see Table 11). It showcased outstanding performance across all key metrics in comparison to other state of the art models. As distance metric L1 distance is used for all models to set the threshold. Here the complete dataset is used and all models are equally optimized using optuna.

In a further evaluation on the simulated smart bridge dataset, ModeConvLaplace demonstrated significant superiority. It achieved the highest scores among all models (see Table 12). Here again L1 distance is used as distance metric to set the threshold and all modells are optimized using optuna equally.

Our findings highlight the efficacy of ModeConv variants in capturing complex spatiotemporal dependencies, especially evident in the ModeConvFast model on the Luxembourg dataset and the ModeConvLaplace model on the smart bridge dataset.

8 Complexity reduction

As demonstrated on both the Luxemburg and the Smart Bridge Dataset, the ModeConv Laplace model exhibits a runtime comparable to ChebConv due to a similar scheme of spectral graph convolution technique based on the normalized Laplacian, while ModeConv Fast has a significantely lower runtime due to its own convolutional filter (see ModeConv Method section).

To comprehend the ChebConv layer [26], it is neccessary to grasp the concept of Spectral Graph Convolution (SGC) and subsequently, the influence of the Chebyshev polynomials K. Let $x(t) : V \rightarrow \mathbb{R}^{N \times T}$ denote a signal within a multivariate time-series, defined at time step t across the graph's nodes, where the hidden state is sequentially calculated for each timestamp t. The graph $G(n, e, w)$ comprises n nodes, e edges, and w weights $\in \mathbb{R}^{N \times N}$, with the neighborhood $N_n = (e_1, \ldots, e_{d_n})^T$ for each node n taken into consideration.

Just Accepted

Table 12. Results for the complete simulated smart bridge dataset: ModeConv with L1 distance against other state of the art models

Model	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
MLP [34]	63.82	12.64	21.1	52.73
VAE [51]	73.87	21.54	33.34	56.98
TCNAE [75]	88.09	45.02	59.48	69.50
TGCN [88]	64.92	12.76	21.31	52.96
STGCN [85]	78.39	23.95	36.57	58.67
MTGNN [82]	67.57	15.19	24.78	53.95
AGCRN [8]	65.87	13.18	21.95	53.19
ModeConvFast	97.4	**90.7**	**93.9**	66.5
ModeConvLaplace	**99.4**	86.5	92.5	**86.8**
ChebConv Defferrard et al. [25]	99.3	55.4	71.1	73.5

Just Accepted

In the case of SGC, N_n is convolved with a degree-specific filter $f \in \mathbb{R}^{d_n}$ via $x(N_n)^T f$. Graphs can then be represented by the Graph Laplacian.

The Graph Laplacian L is symmetric and thus diagonalizable. In $L = U\Lambda U^T$, $U \in \mathbb{R}^{N\times N}$ represents the orthogonal matrix of eigenvectors, and Λ the diagonal matrix of corresponding eigenvalues of L.

This diagonalization facilitates defining the Graph Fourier Transform (GFT) $\hat{f}(t)$ of a signal $f(t) : V \to \mathbb{R}$ on the graph: $\hat{f} = U^T f$ and $f = U\hat{f}$. This GFT transforms a signal to the frequency or spectral domain. When a signal $x(t) : V \to \mathbb{R}^{N\times T}$ is convolved with a filter $F \in \mathbb{R}^{N\times N}$, this spectral filter can be defined as $x(t) * GF := UFU^T x(t)$.

Instead of employing the spectral convolutional filter as introduced, a polynomial approximation of order K is utilized for the filter $F \approx \sum_{k=0}^{K} \theta_k \Lambda^k$ in case of the ChebConv layer, where $\theta \in \mathbb{R}^{K+1}$ is a vector of polynomial coefficients. The convolution then takes the following form:

$$x(t) * GF = UFU^T x(t) \approx U \sum_{k=0}^{K} \theta_k \Lambda^k U^T x(t) \ = \sum_{k=0}^{K} \theta_k U \Lambda^k U^T x(t) \tag{16}$$

Employing Chebyshev polynomials offers two distinct advantages: $T_k(x)$ can be efficiently computed due to its recursive definition, and U and U^T are no longer necessary in the convolution:

$$
\begin{aligned}
x(t) * GF &= UFU^T x(t) \\
&\approx U \sum_{k=0}^{K} \theta_k T_k(\Lambda) U^T x(t) \\
&= \sum_{k=0}^{K} \theta_k U T_k(\Lambda) U^T x(t) \\
&= \sum_{k=0}^{K} \theta_k T_k(U^T \Lambda U) x(t) \\
&= \sum_{k=0}^{K} \theta_k T_k(L) x(t)
\end{aligned}
\tag{17}
$$

Regarding the utilization of Chebyshev polynomials, the time complexity is denoted by $O(K|E|)$, where K denotes the order of the Chebyshev polynomial, and $|E|$ represents the number of edges in the graph (Defferrard et al., 2017). Considering $|E|$ as the number of edges in the graph, the time complexity for ChebConv becomes $O(K|E|)$. For a fully connected graph with n nodes, we have:

$$
|E| = \binom{n}{2} = \frac{n \times (n-1)}{2} \tag{18}
$$

The order K of the Chebyshev polynomial thus plays a significant role in the performance of the ChebConv layer. Increasing the order of the polynomial generally leads to higher approximation accuracy of the spectral filters. A higher-order polynomial can capture more intricate spectral characteristics, allowing for finer-grained analysis of the graph signals. However, this comes at the cost of increased computational complexity. As K grows, the computational burden of computing and applying the polynomial approximation also increases, potentially leading to longer training and inference times, especially in large-scale graph datasets. While a higher-order polynomial offers improved accuracy in capturing spectral features, it may also introduce computational overhead. Therefore, practitioners often need to balance between the desired level of accuracy and the computational resources available.

On the other hand, the time complexity of ModeConv Fast involves computing the Singular Value Decomposition (SVD) of an $m \times n$ matrix, where m is the number of eigenmodes and n is the number of sensors. Assuming that m represents the number of chosen eigenmodes and n represents the number of sensors. Based on the LAPACK implementation used by numpy the time complexity for computing the SVD of the signal x is then given by [43]:

$$
T_{\text{SVD}} = \begin{cases} O(mn^2) & \text{if } m \geq n \\ O(n^3) & \text{if } m < n \end{cases} \tag{19}
$$

Therefore, for a fully connected graph with n nodes, ChebConv has a higher time complexity than ModeConvFast when $K = 5$, which was optimized during training. This comparison is highly dependent on the specified parameters and the structure of the graph. Notably, as the graph densifies with more edge connections, the efficiency of ModeConv Fast accentuates significantly. This is attributed to its adeptness at reducing the matrix size in accordance with the number of eigenmodes. Consequently, in scenarios in which the graph has a large number of edges with a significantly smaller number of eigenmodes compared to sensors, the efficiency gains of ModeConv Fast become increasingly visible. In the conducted additional profiling study, the results indicate that the ModeConv Fast exhibits different speed profiles compared to ChebConv. While the forward pass (`call_impl`)

Just Accepted

of the model in ModeConv Fast becomes faster, the preprocessing in the `process_batch` takes longer. Specifically, the computation of covariances and the execution of SVD contribute to this increased time requirement. This suggests that ModeConv Fast requires additional time for preprocessing to generate the necessary weight matrices and similarity values, whereas ChebConv creates the Laplacian matrix during the forward pass, leading to a lower percentage of total time for preprocessing. The precomputation of preprocessing steps allows one to further enhance the efficiency of ModeConv Fast, which has been implemented.

9 Utilizing ModeConv in Complex Structural Analysis

Since ModeConv allows the modification of input parameters and sensor setup configurations, it can be adapted to different datasets with varying structural and material properties. All other sensor values, e.g. from temperature, can also be fed in as dynamical node features, which is important, as many conventional GNNs are not designed to input additional dynamical features. The ModeConv kernel is able to process complex numbers, for which normal GNN-convolutional kernels are not designed. Therefore, it can process more complex datasets from real-world applications. Bridges can often be modeled as a one-degree-of-freedom system, also known as a single-mass oscillator. In this context, the mass of the structure is a critical parameter that influences its vibrational characteristics. This mass can be accurately known from the construction specifications and remains a fixed input. This value can be changed for other datasets on other bridges, as the mass will not be the same. Nevertheless, we also acknowledge the inherent complexity and non-linearity of real-world structural systems, which often extend beyond the behavior of standard spring-mass-damper models. Real-world systems often do not behave as interconnected spring-mass-damper models. We are therefore very interested in exploring datasets with these nuanced dynamic responses in the future.

10 Discussion

In this work, we explored the application of the ModeConv layer within a Graph Autoencoder framework, demonstrating its capability to detect anomalies through changes in eigenmodes. While our experiments focused on anomaly detection, the versatility of the ModeConv layer extends to other tasks such as classification and regression. These potential applications represent promising directions for future research and experimentation to further explore the capabilities and adaptability of the ModeConv methodology in various contexts.

11 Conclusion

ModeConv offers a novel approach that integrates graph-based representation, covariance and CSD computation, modal decomposition, and graph convolutions for continuous condition monitoring of structures. ModeConv is built upon existing techniques and concepts such as Spectral Graph Convolutions and Fourier/Laplace transforms as the base. As novelty, it uses a novel complex convolutional filter to learn the spatial and temporal patterns in the extracted features and leverages the modal coordinates as inputs to this convolution. This combination enables the detection of subtle structural changes and facilitates accurate damage detection. It further has potential applications in a wide range of domains beyond structural health monitoring, such as speech recognition, audio data, mechanics, quantum physics or simplified in anything that vibrates.

This paper introduces a novel extension to Spectral Graph Convolutions by modulating natural modes and retaining only the relevant natural frequencies while eliminating irrelevant frequencies. The main advantages claimed for ModeConv are better results on anomaly detection tasks related to physical material properties and improved runtime and parameter efficiency.

In reference to the first claim, the ModeConv model, particularly the ModeConvFast variant, achieved mostly higher F1 scores, precision, and recall compared to other models (ChebConv, MtGNN, AGCRN) on the two large-scale datasets. This indicates that ModeConv performs well in detecting anomalies in sensor data based on

Just Accepted

the physical properties of the material. In reference to the second claim, the ModeConvFast model achieved the best performance in terms of F1 score, balanced accuracy, and AUC in the Luxemburg dataset, while also having the lowest training time per epoch.

Additionally, a comparative study was conducted to examine the impact of different components of the ModeConv layer. The results showed that the combination of using two linear layers instead of Cross Spectral Density calculation and utilizing the Laplacian instead of SVD in the Convolutional block resulted in similar performances of all ModeConv variants.

12 Acknowledgements

We thank the German Federal Highway Research Institute (Bundesanstalt für Straßenwesen = BASt) for providing us with the smart bridge dataset as well as the simulated damages dataset in cooperation with Prof. Freundt and IB Freundt. Furthermore we thank the German Federal Ministry for Digital and Transport (BMDV) for financing the P-BIM project. Last but not least we are expressing our heartfelt thanks to Prof. Maas and Khatereh Dakhili from the University of Luxembourg for sharing the *Luxemburg dataset* with us. We also thank Prof. Bodo Rosenhahn for his valuable insights.

References

[1] AI Abu-Tair, C McParland, JF Lyness, and A Nadjai. 2002. Predictive models of deterioration rates of concrete bridges using the factor method based on historic inspection data. In *9th International conference on durability of building materials and components (DBMC), Brisbane, Australia.*

[2] Duzgun Agdas, Jennifer A Rice, Justin R Martinez, and Ivan R Lasa. 2016. Comparison of visual inspection and structural-health monitoring as bridge condition assessment methods. *Journal of Performance of Constructed Facilities* 30, 3 (2016), 04015049.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. arXiv:1907.10902 [cs.LG]

[4] Iván Gómez Araújo and Jose Elias Laier. 2014. Operational modal analysis using SVD of power spectral density transmissibility matrices. *Mechanical Systems and Signal Processing* 46, 1 (2014), 129–145. doi:10.1016/j.ymssp.2014.01.001

[5] Hugo Bachmann. 1995. *Vibration problems in structures: practical guidelines.* Springer Science & Business Media.

[6] Subramaniam Bagavathiappan, Barid Baran Lahiri, T Saravanan, John Philip, and Tammana Jayakumar. 2013. Infrared thermography for condition monitoring–A review. *Infrared Physics & Technology* 60 (2013), 35–55.

[7] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. doi:10.48550/ARXIV.2007.02842

[8] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. arXiv:2007.02842 [cs.LG] https://arxiv.org/abs/2007.02842

[9] Chris Beards. 1996. *Structural vibration: analysis and damping.* Elsevier.

[10] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. 2021. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Comput. Surv.* 54, 3, Article 56 (apr 2021), 33 pages. doi:10.1145/3444690

[11] Hans Anton Buchholdt and Shodja Edin Moossavi Nejad. 1997. *Structural dynamics for engineers.* Thomas Telford London.

[12] E Caetano, S Silva, and J Bateira. 2011. A vision system for vibration monitoring of civil engineering structures. *Experimental Techniques* 35 (2011), 74–82.

[13] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Conguri Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2021. Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting. doi:10.48550/ARXIV.2103.07719

[14] Silvia Cateni, Valentina Colla, Marco Vannucci, Jesus Aramburo, and Antonio Ramirez Trevino. 2008. Outlier detection methods for industrial applications. *Advances in Robotics, Automation and Control* (2008), 265–282.

[15] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *International conference on machine learning*. PMLR, 1725–1735.

[16] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf

[17] Haksoo Choi, Sukwon Choi, and Hojung Cha. 2008. Structural health monitoring system based on strain gauge enabled wireless sensor nodes. In *2008 5th International Conference on Networked Sensing Systems*. IEEE, 211–214.

Just Accepted

[18] William T Cochran, James W Cooley, David L Favin, Howard D Helms, Reginald A Kaenel, William W Lang, George C Maling, David E Nelson, Charles M Rader, and Peter D Welch. 1967. What is the fast Fourier transform? *Proc. IEEE* 55, 10 (1967), 1664–1674.
[19] Dragos M Cvetkovic et al. 1980. Spectra of graphs. Theory and application. (1980).
[20] KY Dai and GR Liu. 2007. Free and forced vibration analysis using the smoothed finite element method (SFEM). *Journal of Sound and Vibration* 301, 3-5 (2007), 803–820.
[21] Alan Davies. 2012. *Handbook of condition monitoring: techniques and methodology.* Springer Science & Business Media.
[22] Eliana Henriques De Brito and Jack Hale. 1982. The damped elastic stretched string equation generalized: existence, uniqueness, regularity and stability. *Applicable Analysis* 13, 3 (1982), 219–233.
[23] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. 2000. The mahalanobis distance. *Chemometrics and intelligent laboratory systems* 50, 1 (2000), 1–18.
[24] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. (2016). doi:10.48550/ARXIV.1606.09375
[25] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).
[26] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2017. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. arXiv:1606.09375 [cs.LG]
[27] Dieter Dinkler. 2020. *Bewegungsgleichungen für Stabtragwerke.* Springer Fachmedien Wiesbaden, Wiesbaden, 96–119. doi:10.1007/978-3-658-31845-1_11
[28] Gustav Doetsch. 2013. *Handbuch der Laplace-Transformation: Band I: Theorie der Laplace-Transformation.* Vol. 14. Springer-Verlag.
[29] FLM Dos Santos, B Peeters, J Lau, W Desmet, and LCS Góes. 2014. An overview of experimental strain-based modal analysis methods. In *Proceedings of the International Conference on Noise and Vibration Engineering (ISMA), Leuven, Belgium.* 2453–2468.
[30] Frank J Fahy and Paolo Gardonio. 2007. *Sound and structural vibration: radiation, transmission and response.* Elsevier.
[31] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-Temporal Graph ODE Networks for Traffic Flow Forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21).* ACM. doi:10.1145/3447548.3467430
[32] Ingenieurbüro Prof. Dr. U. Freundt. 2020-09-30. Erzeugung künstlicher Messdaten einer geschädigten Brücke. https://www.intelligentebruecke.de/DE/Publikationen/Fachveroeffentlichungen/Ingenieurbu/Downloads/kuenstliche-messdaten.pdf
[33] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining.* 1416–1424.
[34] Matt W Gardner and SR Dorling. 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment* 32, 14-15 (1998), 2627–2636.
[35] Adnan Gavili and Xiao-Ping Zhang. 2017. On the shift operator, graph frequency, and optimal filtering in graph signal processing. *IEEE Transactions on Signal Processing* 65, 23 (2017), 6303–6318.
[36] Henri P Gavin. 2014. Structural element stiffness, mass, and damping matrices. *CEE 541. Structural Dynamics* (2014).
[37] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1263–1272. https://proceedings.mlr.press/v70/gilmer17a.html
[38] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT Press, Cambridge, MA, USA. http://www.deeplearningbook.org.
[39] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* 2 (2005), 729–734 vol. 2.
[40] Benjamin A Graybeal, Brent M Phares, Dennis D Rolander, Mark Moore, and Glenn Washer. 2002. Visual inspection of highway bridges. *Journal of nondestructive evaluation* 21, 3 (2002), 67–83.
[41] Karl-Heinz Hapel and Karl-Heinz Hapel. 1990. Zufallsschwingungen linearer Systeme. *Festigkeitsanalyse dynamisch beanspruchter Offshore-Konstruktionen* (1990), 160–192.
[42] Douglas M Hawkins. 1980. *Identification of outliers.* Vol. 11. Springer.
[43] ER Henry and J Hofrichter. 1992. [8] Singular value decomposition: Application to analysis of experimental data. In *Methods in enzymology.* Vol. 210. Elsevier, 129–192.
[44] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A Fast Learning Algorithm for Deep Belief Nets. *Neural computation* 18 (08 2006), 1527–54. doi:10.1162/neco.2006.18.7.1527
[45] Shuguang Huang and Joseph M Schimmels. 2002. The duality in spatial stiffness and compliance as realized in parallel and serial elastic mechanisms. *J. Dyn. Sys., Meas., Control* 124, 1 (2002), 76–84.
[46] R.A. Irizarry. 2019. *Introduction to Data Science: Data Analysis and Prediction Algorithms with R.* CRC Press. https://books.google.de/books?id=62K-DwAAQBAJ
[47] Sanja S Ivanovic, Mihailo D Trifunac, and MI Todorovska. 2000. Ambient vibration tests of structures-a review. *ISET Journal of earthquake Technology* 37, 4 (2000), 165–197.

Just Accepted

[48] Chenhao Jin, Shinae Jang, Xiaorong Sun, Jingcheng Li, and Richard Christenson. 2016. Damage detection of a highway bridge under severe temperature changes using extended Kalman filter trained neural network. *Journal of Civil Structural Health Monitoring* 6 (2016), 545–560.
[49] Ming Jin, Guangsi Shi, Yuan-Fang Li, Qingsong Wen, Bo Xiong, Tian Zhou, and Shirui Pan. 2023. How Expressive are Spectral-Temporal Graph Neural Networks for Time Series Forecasting?
[50] Ming Jin, Yu Zheng, Yuan-Fang Li, Siheng Chen, Bin Yang, and Shirui Pan. 2023. Multivariate Time Series Forecasting With Dynamic Graph Neural ODEs. *IEEE Transactions on Knowledge and Data Engineering* 35, 9 (Sept. 2023), 9168–9180. doi:10.1109/tkde.2022.3221989
[51] Diederik P. Kingma and Max Welling. 2019. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning* 12, 4 (2019), 307–392. doi:10.1561/2200000056
[52] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. doi:10.48550/ARXIV.1609.02907
[53] Jingxiao Liu, Siheng Chen, Mario Bergés, Jacobo Bielak, James H. Garrett, Jelena Kovačević, and Hae Young Noh. 2020. Diagnosis algorithms for indirect structural health monitoring of a bridge model via dimensionality reduction. *Mechanical Systems and Signal Processing* 136 (2020), 106454. doi:10.1016/j.ymssp.2019.106454
[54] Man Liu and Dadiv G Gorman. 1995. Formulation of Rayleigh damping and its extensions. *Computers & structures* 57, 2 (1995), 277–285.
[55] Filipe Magalhães and Álvaro Cunha. 2011. Explaining operational modal analysis with data from an arch bridge. *Mechanical Systems and Signal Processing* 25, 5 (2011), 1431–1450.
[56] Filipe Magalhães, Álvaro Cunha, Elsa Caetano, and Rune Brincker. 2010. Damping estimation using free decays and ambient vibration tests. *Mechanical Systems and Signal Processing* 24, 5 (2010), 1274–1290.
[57] MD Malkauthekar. 2013. Analysis of Euclidean distance and Manhattan distance measure in Face recognition. In *Third International Conference on Computational Intelligence and Information Technology (CIIT 2013)*. IET, 503–507.
[58] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognition* 97 (2020), 107000.
[59] P Kumar Mehta and Richard W Burrows. 2001. Building durable structures in the 21st century. *Concrete international* 23, 3 (2001), 57–63.
[60] Howard E Michel, Abdul Ahad S Awwal, and David Rancour. 2006. Artificial neural networks using complex numbers and phase encoded weights-electronic and optical implementations. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 486–491.
[61] Spiros Papadimitriou and Christos Faloutsos. 2003. Cross-outlier detection. In *International Symposium on Spatial and Temporal Databases*. Springer, 199–213.
[62] Mayuresh Patil. 2000. Decoupled second-order equations and modal analysis of a general nonconservative system. *41st Structures, Structural Dynamics, and Materials Conference and Exhibit*, 1654.
[63] Bart Peeters and Guido De Roeck. 2001. Stochastic system identification for operational modal analysis: a review. *J. Dyn. Sys., Meas., Control* 123, 4 (2001), 659–667.
[64] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).
[65] C. Rainieri and G. Fabbrocino. 2014. *Operational Modal Analysis of Civil Engineering Structures: An Introduction and Guide for Applications*. Springer New York. https://books.google.de/books?id=LzmZngEACAAJ
[66] Edwin P.B. Reynders. 2021. Uncertainty quantification in data-driven stochastic subspace identification. *Mechanical Systems and Signal Processing* 151 (2021), 107338. doi:10.1016/j.ymssp.2020.107338
[67] T. Konstantin Rusch, Benjamin P. Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael M. Bronstein. 2022. Graph-Coupled Oscillator Networks. arXiv:2202.02296 [cs.LG]
[68] Olusegun S Salawu and Clive Williams. 1995. Bridge assessment using forced-vibration testing. *Journal of structural engineering* 121, 2 (1995), 161–173.
[69] Sebastian Schommer. 2017. *Damage detection in prestressed concrete bridges based on static load testing, sagging and modal parameters, using measurements and model updating*. Ph. D. Dissertation. University of Luxembourg, Luxembourg.
[70] M Servin, JC Estrada, and J Antonio Quiroga. 2009. The general theory of phase shifting algorithms. *Optics express* 17, 24 (2009), 21867–21881.
[71] Bartosz Siedziako, Ole Øiseth, and Anders Rønnquist. 2017. An enhanced forced vibration rig for wind tunnel testing of bridge deck section models in arbitrary motion. *Journal of Wind Engineering and Industrial Aerodynamics* 164 (2017), 152–163.
[72] Michael L Stein. 2005. Space–time covariance functions. *J. Amer. Statist. Assoc.* 100, 469 (2005), 310–321.
[73] Gilbert W Stewart. 1993. On the early history of the singular value decomposition. *SIAM review* 35, 4 (1993), 551–566.
[74] Sebastian Tatzko and Martin Jahn. 2019. On the use of complex numbers in equations of nonlinear structural dynamics. *Mechanical Systems and Signal Processing* 126 (2019), 626–635.
[75] Markus Thill, Wolfgang Konen, Hao Wang, and Thomas Bäck. 2021. Temporal convolutional autoencoder for unsupervised anomaly detection in time series. *Applied Soft Computing* 112 (2021), 107751. doi:10.1016/j.asoc.2021.107751

Just Accepted

[76] S Vulli, JF Dunne, R Potenza, D Richardson, and P King. 2009. Time-frequency analysis of single-point engine-block vibration measurements for multiple excitation-event identification. *Journal of Sound and Vibration* 321, 3-5 (2009), 1129–1143.
[77] Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. 2003. Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*. Springer, 91–109.
[78] Yueyang Wang, Ziheng Duan, Yida Huang, Haoyan Xu, Jie Feng, and Anni Ren. 2020. MTHetGNN: A Heterogeneous Graph Embedding Framework for Multivariate Time Series Forecasting. doi:10.48550/ARXIV.2008.08617
[79] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.
[80] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.
[81] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24. doi:10.1109/TNNLS.2020.2978386
[82] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. arXiv:2005.11650 [cs.LG]
[83] Jingke Xi. 2008. Outlier detection algorithms in data mining. In *2008 Second International Symposium on Intelligent Information Technology Application*, Vol. 1. IEEE, 94–97.
[84] Wenxian Yang, Ziqiang Lang, and Wenye Tian. 2015. Condition monitoring and damage location of wind turbine blades by frequency response transmissibility analysis. *IEEE Transactions on Industrial Electronics* 62, 10 (2015), 6558–6564.
[85] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization. doi:10.24963/ijcai.2018/505
[86] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2018. Graph convolutional networks: Algorithms, applications and open challenges. In *Computational Data and Social Networks: 7th International Conference, CSoNet 2018, Shanghai, China, December 18–20, 2018, Proceedings 7*. Springer, 79–91.
[87] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2019. Graph convolutional networks: a comprehensive review. *Computational Social Networks* 6, 1 (2019), 1–23.
[88] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2020. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems* 21, 9 (Sept. 2020), 3848–3858. doi:10.1109/tits.2019.2935152

Received 27 June 2024; revised 1 July 2025; accepted 2 February 2026

Just Accepted