Studia Geotechnica et Mechanica
Volume 46 Issue s1 · pp. 417–441 · DOI: 10.2478/sgem-2025-0020

Studia Geotechnica et Mechanica

Application of machine learning tools in road bridge weigh-in-motion systems

Research Article

Aleksander Mróz*, Tomasz Kamiński, Jan Bień

Faculty of Civil Engineering, Wrocław University of Science and Technology, Wrocław, Poland

Received 13 January 2025; Accepted 13 August 2025

Abstract: This study explores the application of machine learning (ML) techniques in road and bridge weigh-in-motion (B-WIM) systems, focusing on their potential to improve the accuracy and efficiency of load identification processes. The study begins with an introduction to the fundamentals of ML and artificial intelligence, providing an overview of key algorithm families and their applicability in bridge civil engineering. Subsequently, the concept of B-WIM systems is discussed, emphasizing the role of ML in vehicle classification, detection, and load estimation. A novel system architecture is proposed, integrating the You Only Look Once algorithm for real-time vehicle detection and tracking, and autoencoders for analysing the structural response signal to estimate axle loads. Proof-of-concept case studies, implemented within a numerical simulation environment (finite element software SOFiSTiK), are presented to illustrate the feasibility of such approaches. These examples demonstrate key concepts in a simplified form, supporting the proposed methodology without requiring full-scale experimental validation. Results demonstrate the advantages of ML tools in handling dynamic loads and diverse traffic conditions, significantly outperforming traditional methods regarding accuracy and scalability. The study concludes by highlighting the broader implications of using ML in structural monitoring and its potential to revolutionize load identification in bridge engineering.

Keywords: *Machine learning • Bridge weigh-in-motion • Road bridges • Vehicle identification • Load identification*

List of abbreviations

AI	artificial intelligence
B-WIM	bridge weigh-in-motion
CNN	convolution neural network
FAD	free axle detector
FOV	field of view
FRP	fixed reference point
IF	influence function
LIS	load identification system
MDS	mechanical detection system
MFI	moving force identification
ML	machine learning
MSE	mean squared error
NOR	nothing on the road
RMSE	root mean squared error
VAE	variational autoencoder
VIS	vehicle identification system
WIM	weigh-in-motion
YOLO	You Only Look Once

* *E-mail: aleksander.mroz@pwr.edu.pl*

1. Machine learning (ML) in civil engineering

1.1 Fundamentals of ML and artificial intelligence (AI)

AI is an interdisciplinary field of science concerned with creating systems capable of performing tasks that typically require human intelligence. It encompasses processes such as learning, reasoning, decision-making, and processing information from the environment to interpret and respond to it. AI is a broad term covering many distinct and often unrelated techniques, such as ML, natural language

processing, expert systems, image and speech processing, planning, and robotics. Rather than being a single method, AI is a collection of approaches designed to replicate or support different aspects of human intelligence. ML is a field of computer science focused on methods that enable computers to learn independently from available data, without the need for traditional, manual programming of all rules. In ML, algorithms use input data to automatically detect patterns that can be then used for predictions or other forms of data analysis. The difference between the mechanism of traditional programming or problem-solving and the mechanism utilizing ML is presented in Figure 1.

ML is thus an area of AI that enables computers to learn from experience, improve their actions, and make decisions without the need to create complex programs with rules that account for all possible cases, which is usually impossible to achieve.

ML covers a very broad field, comprising a wide range of algorithms that can differ significantly from one another. In the context of ML development and applications, it is essential to define fundamental categories, which allow for the classification of methods based on specific features and requirements. This basic categorization considers the level of human supervision in the learning process.

According to some publications [1,2], categories based on the level of human supervision include:

- Supervised learning,
- Unsupervised learning,
- Semi-supervised learning,
- Reinforcement learning (RL).

The development of ML has progressed through various stages, encompassing multiple categories of algorithms, each contributing new capabilities for analysis and prediction.

Table 1 presents the main families of ML algorithms along with selected representatives from each group and a short characteristic. ML includes several groups of algorithms. Each group uses a different way to learn from data or make predictions. These groups are called algorithm families. The same algorithm can often be used in different ways, depending on the task and how it is trained.

Table 1 shows the main algorithm families, with examples and short descriptions.

ML is a field of science whose roots reach back long before the era of modern computers. The overview in Table 1 clearly shows that ML is built on a solid foundation of classical statistical methods, developed as early as the late nineteenth century. Today's popular neural networks represent only a part of this long- standing tradition. The success of modern ML applications has been made possible by decades of evolution in various approaches and algorithms.

1.2 Areas of ML applications in civil engineering

ML is particularly effective in areas with high problem complexity that exceeds the capabilities of traditional analytical methods.

Key applications of ML include the following:

- problems requiring dynamic adaptation,
- complex analytical challenges,
- support in data analysis.

In civil engineering applications ML can be applied to general types of tasks, as illustrated with examples and corresponding algorithms in Table 2.

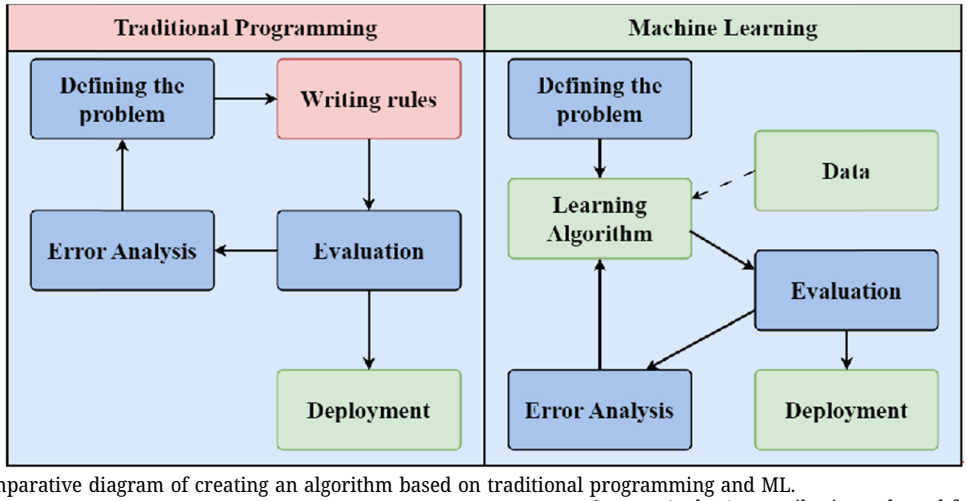

Figure 1. Comparative diagram of creating an algorithm based on traditional programming and ML.
Source: Author's contribution; adapted from Géron [1].

Algorithm family	Selected algorithms	Description
Linear methods	Linear Regression	Introduced as early as the nineteenth century by Francis Galton, formalized as a statistical model by Karl Pearson at the beginning of the twentieth century
	Logistic Regression	Developed by David Cox in the 1950s
	Ridge Regression	Introduced in the 1970s by Hoerl and Kennard
	Lasso Regression	Developed by Robert Tibshirani in 1996
Decision trees and variants	Decision Trees	Developed from the 1960s, formally defined in the 1980s by Leo Breiman (CART algorithm – Classification and Regression Trees)
	Random Forests	Introduced by Leo Breiman in 2001 as a combination of multiple decision trees to improve model accuracy
Support vector machines	Support vector machines	Developed in the 1990s by Vladimir Vapnik and his collaborators; became popular in the second half of that decade
Probabilistic methods	Naïve Bayes Classifier	Originates from the eighteenth century, gained importance in the 1960s as a probabilistic method for text classification
	Hidden Markov Models	Developed by Leonard E. Baum in the 1960s; widely used in the 1970s and 1980s for sequence analysis, e.g. in speech recognition
Dimensionality reduction	Principal Component Analysis	Introduced by Karl Pearson in 1901
	Linear Discriminant Analysis and Quadratic Discriminant Analysis	Linear Discriminant Analysis introduced by Ronald Fisher in 1936; Quadratic Discriminant Analysis (QDA) developed later as a statistical classification method
Clustering	K-means	Developed by Stuart Lloyd in 1957 (published in 1982); widely used for data clustering
	Density-Based Spatial Clustering of Applications with Noise	Density-Based Spatial Clustering of Applications with Noise, introduced by Martin Ester and colleagues in 1996 as a method for grouping spatial data
Artificial neural networks	Feedforward Neural Network	Early neural network model, developed in the 1950s, particularly by researchers such as Warren McCulloch and Walter Pitts, who proposed a theoretical model in 1943
	Self-Organizing Map	Developed by Teuvo Kohonen in the 1980s; widely used for data visualization and clustering
	Radial Basis Function Network	Introduced by Broomhead and Lowe in the 1980s
	Recurrent Neural Network	Developed in the 1980s, gaining popularity in the 1990s with architectures like Long Short-Term Memory (LSTM) proposed by Hochreiter and Schmidhuber in 1997
	Convolutional Neural Network (CNN)	Based on convolution layers, developed by Yann LeCun in the 1980s, gaining popularity in the late 2000s
	Autoencoder	Developed in the 1980s, with significant advancements in the 2000s alongside deep learning
	Generative Adversarial Network	Introduced by Ian Goodfellow and collaborators in 2014; popular for generating synthetic data
	Transformer	Introduced by Vaswani and collaborators in 2017, revolutionized natural language processing by eliminating the need for sequential processing
	Graph Neural Network	Developed in the early twenty-first century, gaining popularity after 2010; used for processing data in graph form

Table 1. Overview of ML algorithm families.

Source: Author's contribution; concept based on Géron [1].

Task type	Description	Exemplary applications in civil engineering	Possible ML algorithms
Regression	Used to predict continuous values	– Regression models predict total construction costs based on data [3,4] – Application of ML models to predict the compressive strength of new modern structural materials [5]	***Linear Methods:*** *– Linear Regression* *– Multiple Regression* ***Neural Networks:*** *– Multilayer Perceptron*
Classification	Useful for assigning categories to specific observations, such as material defect classification	Automatic defect detection and segmentation in pulsed thermography data for defect detection in materials such as steel, plexiglass, and carbon fibre-reinforced polymer [6]	***Probability-Based Methods:*** *– Naive Bayes Classifier* ***Support Vector Machines:*** *– Support Vector Machines* ***Neural Networks:*** *– Convolutional Neural Networks for images*
Clustering	Used for grouping data based on similarity	Fault detection in industrial devices based on the analysis of text data from service reports using Word2Vec, autoencoders and K-means clustering [7]	***Clustering Algorithms:*** *– DBSCAN* ***Neural Networks:*** *– Self-Organizing Maps*
Anomaly detection	Used to identify deviations from norms, which can be critical in structural monitoring and anomaly detection	Autoencoders for anomaly detection in structural health monitoring [8]	***Probability-Based Methods:*** *– Naive Bayes* ***Support Vector Machines:*** *– One-Class Support Vector Machines* ***Neural Networks:***

(Continued)

Table 2: continued

Task type	Description	Exemplary applications in civil engineering	Possible ML algorithms
Data denoising	ML algorithms help to eliminate noise or interference from data such as images or signals, particularly in analysing raw structural data for better interpretation	GPR data cleaning: The CFFM- ESAM-Res-UNet deep network, combining contextual fusion and spatial attention modules, effectively removes noise in GPR data and enables precise subsurface imaging using reverse time migration [9,10]	– *Autoencoders (especially denoising variants)* ***Neural Networks:*** – *Autoencoders (especially denoising variants)*
Dimensionality reduction	Used to reduce the number of variables in large datasets, facilitating visualization and speeding up computation	Material analysis data reduction: Prediction of the properties of building materials (thermal, mechanical, and optical) and optimization of their production processes [11]	***Reduction Algorithms:*** – *Principal Component Analysis, Linear Discriminant Analysis* ***Neural Networks:***
Time series analysis	Predicting future values based on historical data	Predict the soil liquidity index and classify the soil type based on sequences of CPTU test measurements [12]	– *Autoencoders* ***Neural Networks:*** – *Recurrent Neural Networks*
RL	Particularly useful in controlling systems requiring decision-making	Construction site management: Application of RL in Supply Chain Management [13]	– *LSTM* ***Neural Networks:*** – *Deep Q-Network* – *Actor-Critic Methods*

Table 2. General types of tasks in civil engineering being solved by ML algorithms.

Source: Author's contribution; concept based on Géron [1].

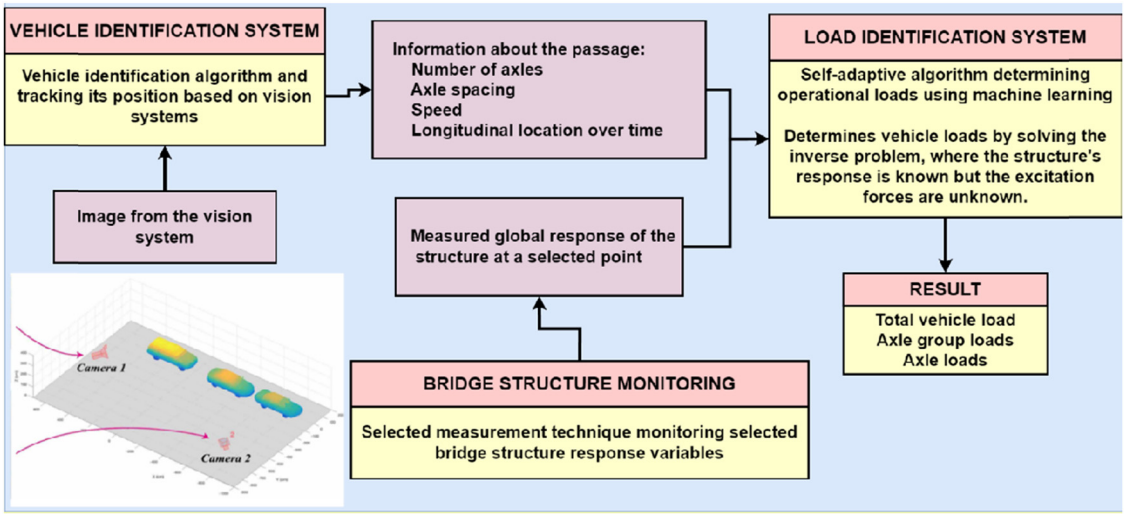

Figure 2. General concept of the proposed B-WIM system structure.

Source: Author's contribution.

Table 2 presents the typical types of tasks faced by ML algorithms in civil engineering. Most real-world problems can be categorized into one of the presented groups. Each of these task types imposes different requirements on the models, which necessitates the use of diverse algorithms. In practice, artificial neural networks are gaining increasing popularity due to their flexibility and ability to adapt their architecture to the specific nature of a problem, making them capable of effectively handling virtually all the listed task types. However, the choice of an appropriate algorithm depends on the specific analytical needs and the characteristics of the data being analysed. While neural networks, especially deep architectures, provide high flexibility in modelling nonlinear and multivariate relationships, they are not universally superior. In engineering contexts where data are scarce or model interpretability is essential, alternative supervised learning algorithms, such as Support Vector Machines or Decision Trees, can offer better generalization performance and easier deployment. This observation is supported in the study by Farrokhi and Rahimi [14] where Support Vector Machines with a Gaussian kernel outperformed artificial neural networks, decision trees, and naive Bayes models. In this study, particular attention is paid to the application of ML methods in bridge weigh-in-motion (B-WIM) systems.

1.3 Potential applications of ML in B-WIM systems

The identification of overloaded vehicles is a topic that has been extensively analysed but has not been comprehensively solved yet. Research conducted by Gdańsk University of Technology [15] shows that the share of overloaded vehicles in road traffic in Poland ranges from 14 to 23%, while another publication [16] indicates that the average share of overloaded vehicles is 18.5%.

The traffic of overloaded vehicles accelerates the degradation processes of all transport infrastructure elements, including bridge structures. According to Rys [15], overloaded vehicles cause from 35% to as much as 70% of pavement structure damage. Additionally, damage may occur to invisible underground networks, such as water or electrical networks. The issue of overloaded vehicle traffic affects all technical classes of roads.

The general concept of a proposed new B-WIM system, shown in Figure 2, is focused on the main goal: developing an algorithm capable of efficiently determining operational live loads.

The figure includes a schematic diagram showing both vehicle identification system (VIS) and load identification system (LIS) modules, indicating how vehicle detection data feed the load identification algorithms. In the proposed B-WIM system, the VIS is the main element responsible for determining the fundamental parameters of a vehicle, such as number of axles, axle spacing, speed, and longitudinal location over time. Vision systems can be used for this purpose, recording real-time vehicle passage to ensure continuous data collection. The second element of the system is the monitoring of the bridge structure response, implemented using selected measurement techniques. The system's task is to record the bridge structure response to the passage of a vehicle.

The critical component of the system is the LIS algorithm that determines load pressures, using ML techniques that are self-adaptive, which means that the algorithm improves its performance autonomously when it is operating. Knowing the vehicle's configuration, based

on data from the VIS and the signal of the bridge response, the algorithm determines the vehicle's total weight.

1.4 Research objectives

The main objective of this article is to provide a basic classification and structural overview of traditional B-WIM systems based on a comprehensive literature review. Additionally, the study introduces and discusses the potential of integrating ML mechanisms into B-WIM systems to enhance their performance in operational load monitoring tasks.

The article focuses on the application of ML methods in vehicle identification and load estimation processes. In addition to the theoretical analysis, this article presents simplified implementations of selected system components, including vehicle detection, and weight identification, using basic computational models and computer simulations. These implementations aim to illustrate and promote the proposed concept rather than deliver a fully operating system. The purpose of this work is to demonstrate the feasibility of the approach and to encourage further research and development, recognizing that creating a complete solution would require greater resources and collaborative effort.

2. Family of weigh-in-motion (WIM) systems

2.1 Road WIM systems

The primary existing tools for automatically measuring axle loads are WIM systems. The first concept of vehicle parameters identification using sensors embedded in the road surface was published in 1952 in the article "Weighing Vehicles in Motion" [17], which described the concept of an electronic device capable of measuring individual axle loads, distances between axles, and detecting the speed of moving vehicles. Case study of the first WIM system implementation is presented in Figure 3.

WIM systems have been developed over many years, and the current state of knowledge on these systems is presented in the series of articles [18–21], which cover a review and classification of WIM systems. In current implementations, the actual measurement result from a WIM system is the instantaneous axle load of a vehicle passing over the measuring device. This poses a significant issue, as moving vehicles experience additional forces from factors such as the inertia of sprung and unsprung masses, aerodynamic effects, road surface irregularities, driving techniques, and vehicle suspension parameters.

In the study by Burnos [21], key factors affecting the accuracy of WIM systems, such as vehicle dynamics, pavement type, sensor placement, and temperature sensitivity, were identified and methods to mitigate their impact were proposed.

2.2 B-WIM systems

An alternative to road-embedded scales is a bridge-based load monitoring system, known in the literature as the B-WIM system, which utilizes the bridge superstructure to identify parameters of vehicles traveling over it, based on the history of interaction effects.

This article presents a general classification and structure of traditional B-WIM systems, based on a literature review,

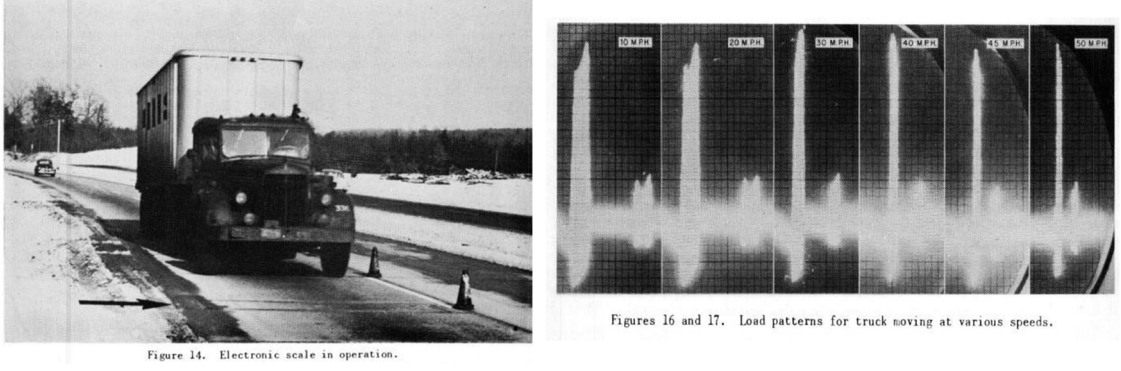

Figure 14. Electronic scale in operation.

Figures 16 and 17. Load patterns for truck moving at various speeds.

Figure 3. The passage of a tractor-trailer with a semi-trailer over a measuring device embedded in the pavement (on the left); The response of the measuring device to the passage of the vehicle at different speeds (on the right).

Source: data from [17].

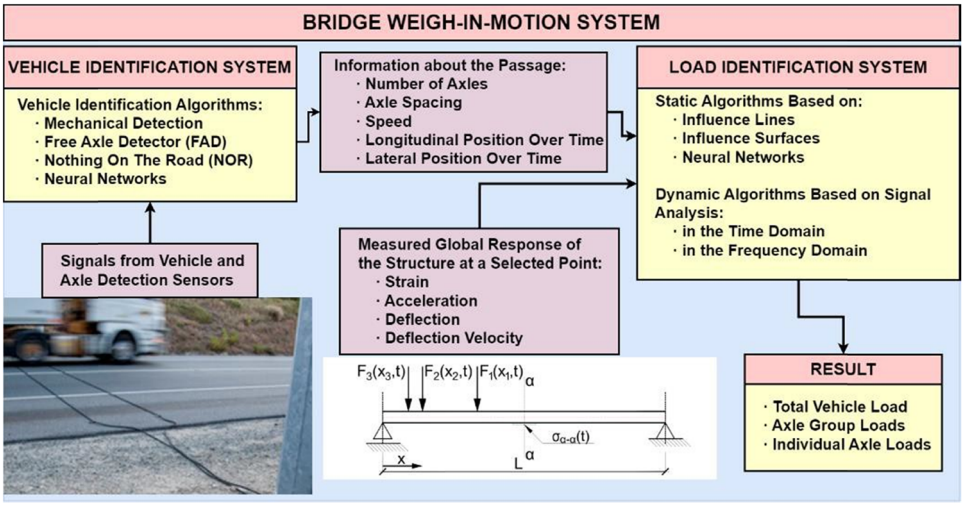

Figure 4. Diagram of a typical B-WIM system.

Source: Author's contribution.

and introduces the concept of a B-WIM operational load monitoring system enhanced with ML mechanisms.

The first concept of a B-WIM system was proposed by Moses in 1979 [22]. Unlike WIM systems, which measure the instantaneous axle load on sensors, B-WIM systems rely on the response history of the bridge structure during vehicle passage, allowing for more accurate vehicle mass estimation. B-WIM systems have significant potential and advantages over other solutions, as they are more resistant to damage. The sensors are installed on the underside of the bridge girders or other structural elements, protecting them from direct weather exposure and contact with vehicles [23]. The installation of B-WIM systems is safe and does not require road closures during installation work. Additionally, B-WIM systems provide a more accurate estimation of vehicle mass than traditional WIM systems.

Figure 4 shows a general framework for typical architecture of B-WIM systems based on different techniques and algorithms.

The core component of the B-WIM systems is VIS, which detects the presence of a vehicle on the bridge, determines its geometry and motion parameters, and then transmits these data to LIS.

There are three traditional types of VIS:

- **Mechanical detection system (MDS)**: This basic solution consists of a pair of pneumatic sensors arranged parallel across the entire road width just before the bridge structure. The passage of a vehicle axle generates a pressure change within the measurement system, which is a measurable quantity [24].
- **Free axle detector (FAD)**: This system uses sensors mounted at selected locations on the underside of the bridge girder or other structural element to detect vehicle passage. Sensors are typically placed in positions where a distinct change in the measured value occurs [25].
- **Nothing on the road (NOR)**: This system generalizes FAD by using not only local deformations, such as those in the web near the supports, but also signals from the global structural response [26].

Vehicle identification and position tracking during its passage over a bridge structure are the key issues for the correct functioning of the entire system. The choice of vehicle identification technique is case-specific due to the varying requirements imposed on B-WIM systems.

Table 3 summarizes the basic properties of the presented VIS, including classification criteria, measurement techniques used, potential outputs, distinctive features, and literature references.

For short-term studies on small structures with low traffic volumes, an MDS can be used. This system reliably detects axles and determines their speed, provided that vehicles move at sufficient distance from each another. However, it is prone to mechanical damage and characterized by low durability. If long-term monitoring is planned on a structure with moderate traffic intensity, FAD or NOR systems should be used. Since the sensors are located beneath the bridge deck, they offer significantly higher durability, though at the cost of employing more complex and less certain methods of data interpretation.

Algorithm	Typical sensor location	Results				Intended use	Literature references
		Axle configuration	Velocity	Longitudinal location	Transverse location		
MDS	In the surface in front of the facility	●	●	○		Short-term monitoring on low- traffic facilities for low-accuracy systems	[22,24,27,28]
FAD	Over support points or in points that give a clear answer	●	●	○	○	Continuous monitoring on medium- traffic facilities for systems with good accuracy	[25,26,29]
NOR	Most often in the middle of the span	●	●	○	○	Continuous monitoring on medium- traffic facilities for systems with good accuracy	[26,30–32]

Explanation of symbols: usefulness (●) – high, (○) – medium, (0 – low.
Table 3. Classification of vehicle identification algorithms in B-WIM systems.

Source: Author's contribution,

LIS (Figure 4) uses vehicle data and structural response to estimate vehicle's mass and axle loads, employing two approaches:

- **Quasi-static**: This approach treats the passage as quasi-static, assuming that the instantaneous axle load may fluctuate, but over time, the average value will correspond to the static load. A representative of this category is Moses algorithm [22], which was developed for beam-slab bridge structures.
- **Dynamic**: This approach considers the passage as a dynamic phenomenon, using equations of motion for continuous or discrete mathematical models to replicate and determine the applied forces.

Quasi-static methods based on Moses concept are widely used due to their ease of implementation and the lack of need for representative bridge models. These methods rely on influence functions (IFs), which describe how a measured static response varies depending on the position of a calibration vehicle with known parameters. Depending on the approach, these functions can take the form of one-dimensional (1D) or two-dimensional (2D) IFs. These methods perform well in estimating the total vehicle mass but face challenges in determining axle or axle group loads, as they do not account for the dynamic nature of the passage. These techniques are most commonly used for beam structures with simple and short-span geometries. However, they can also be applied to determine operational loads on complex structures, such as cable-stayed bridges [33].

Dynamic methods, also known as moving force identification (MFI), consider the dynamic nature of vehicle passage over a bridge. They rely on a representative model, either continuous or discrete, to solve the inverse problem, allowing for a more accurate estimation of axle loads. However, implementing such systems on real bridges presents significant challenges, as real structures are not idealized mathematical models. Although numerous studies focused on improving the accuracy and expanding the applicability of MFI algorithms, most of these investigations (e.g. [34–38]) are still based on virtual models and controlled laboratory conditions. This highlights the need for further research to refine these methods and adapt them for practical deployment in commercial B-WIM systems.

ML-based methods offer the greatest potential by combining elements of both approaches, enabling precise estimation of vehicle mass and axle loads. With their self-improvement capabilities and the ability to account for multiple parameters simultaneously, these algorithms are flexible and can be applied to both quasi-static and dynamic analyses.

3. ML tools in VISs

3.1 Available algorithms

The algorithm in the proposed VIS performs two separate tasks:

- **Classification**: The primary task is to analyse each frame from the vision camera and then detect and classify objects appearing on it. In this system, the desired objects are passenger cars, delivery vehicles, and trucks.
- **Location determination**: The second task is to determine the location of the object on the bridge, i.e. in a three-dimensional space.

The primary task of object classification in an image can be accomplished using several algorithms, such as

- Haar Cascades [39],
- Histogram of Oriented Gradients (HOG) [40],
- Region Convolutional Neural Networks (R-CNN) [41],
- Single Shot MultiBox Detector (SSD) [42],
- You Only Look Once (YOLO) [43],
- Region-based Fully Convolutional Network (R-FCN) [44].

Many methods can be used for classification, ranging from classical algorithms such as Haar Cascades or HOG to modern techniques based on deep neural networks, such as the R-CNN family, SSD, YOLO, or R-FCN. Classical algorithms, while computationally efficient, have limitations when dealing with objects of a complex geometric structure and a varying perspective, which affects detection accuracy. Modern algorithms based on CNN offer higher precision, the ability to detect a variety of objects, and speed suitable for real-time applications. However, their use often comes with higher computational requirements. The optimal choice of algorithm therefore depends on the specific task, particularly the requirements for processing speed and detection accuracy.

Among the methods presented, the YOLO algorithm was selected for further analysis due to its optimal balance between detection accuracy and processing speed, which enables effective real-time performance even on hardware with limited computational capabilities. YOLO allows for simultaneous detection and classification of multiple objects within an image, which is particularly important for vehicle traffic analysis. It also enables tracking of object positions across consecutive frames, as detailed in Section 3.2 of this study.

3.2 Example of application

Figure 5 shows two photos with an implementation of a VIS for determining its position. Initially, an image recorded with a smartphone camera at John Paul II Square in Wrocław was used. The YOLOv4 [43] model was employed for image analysis, as it offers high accuracy while maintaining high processing speed.

Version 4 of the algorithm was chosen because it can be trained on consumer-grade graphics cards. YOLO (especially the newer versions of the V8) can detect partially occluded objects, especially when trained on datasets that include such examples. The algorithm works globally across the entire image, learning local features (e.g. edges, lights, contours) that allow it to recognize vehicles even when they are not fully visible. However, its performance decreases in more complex scenarios, such as when two vehicles are closely spaced, one blocks another (e.g. a truck obscuring a car), or the image quality is reduced due to poor lighting or motion blur. To address these challenges in real-world systems, a multi-camera setup is typically required, often combined with additional sensors like LiDAR or radar. The goal here is to demonstrate a basic yet functional methodology for vehicle detection and position tracking under constrained experimental conditions.

It is also possible to track selected vehicle elements, such as wheels. The location of the tracked vehicle can be

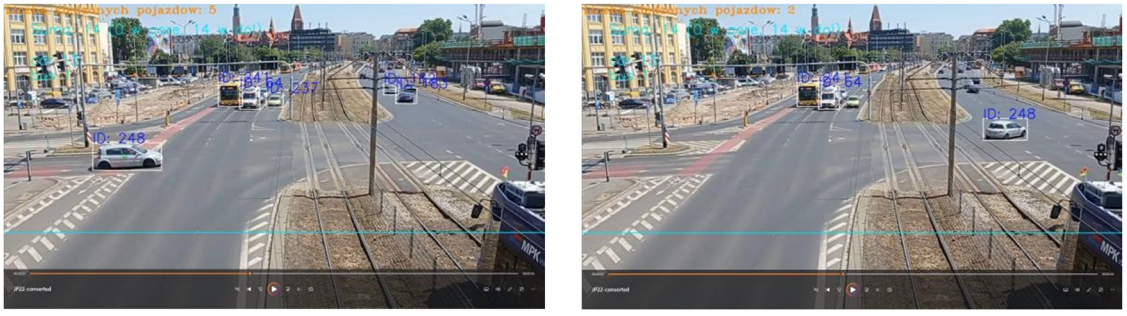

Figure 5. Identification and tracking of vehicles based on camera image.

Source: Author's contribution.

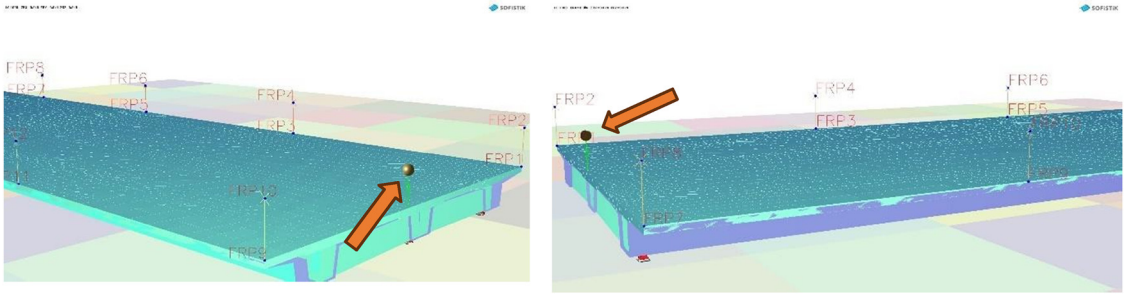

Figure 6. Screenshots of a model mass passage from two different perspectives.

Source: Author's contribution.

saved as a history of the positions of the centre of the bounding box (marked in the image) in the *X*–*Y* space. An image from a single camera is a two-dimensional array containing colour values for individual pixels. However, based on an image from a single camera, it is not possible to accurately determine the position of a point in 3D space. With images from two or more cameras observing the same object from different perspectives, and assuming the presence of fixed reference points (FRPs), it is possible to precisely determine its location in space using triangulation [45].

To illustrate this concept, a simple 3D bridge model was created in the SOFiSTiK software, on which a suspended mass point moves. The mass point travels along an axis parallel to the *X*-axis of the bridge, offset by 1 m to the left. The mass point is located 1.5 m above the road level and moves at a speed of 25 m/s. Two separate videos showing the motion of the mass from different perspectives were recorded at a frequency of 200 Hz. Figure 6 shows the mass point and the adopted reference points (FRP) for the structure for which the *X*, *Y*, *Z* coordinates are known.

Initially, simple classifiers based on Hough circle transform [46] were used to determine the location and centroid of the mass point in each video frame. Subsequently, using triangulation techniques and relying on the known reference points on the structure, the location of the tracked mass point was determined during transit. Several factors affect estimation accuracy, including the field of view (FOV), focal length, distortion (barrel/pincushion), and camera position/angle. FOV influences depth accuracy, focal length impacts magnification, and distortion causes angular inaccuracies. To accurately determine the location of points in 3D space, camera calibration is necessary, accounting for the elimination of distortion effects, and allowing for accurate representation of real distances. In the presented example, screenshots of the simulated passage in the program were used. For this reason, only the camera angles were considered. Figure 7 illustrates the general concept of determining 3D location based on 2D images using triangulation.

Using the algorithm described by Hartley and Zisserman [48], images recorded from different perspectives were processed, and the location of the tracked object during transit was determined. Since everything took place in a simulated environment, the exact position of the mass over time is known, allowing us to

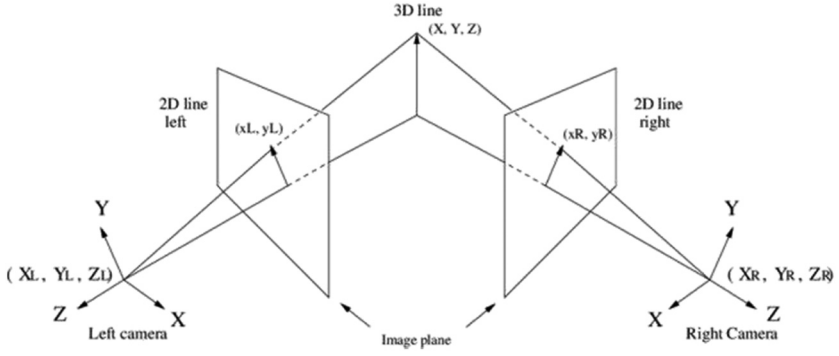

Figure 7. Stereo triangulation from a pair of cameras.

Source: data from [47].

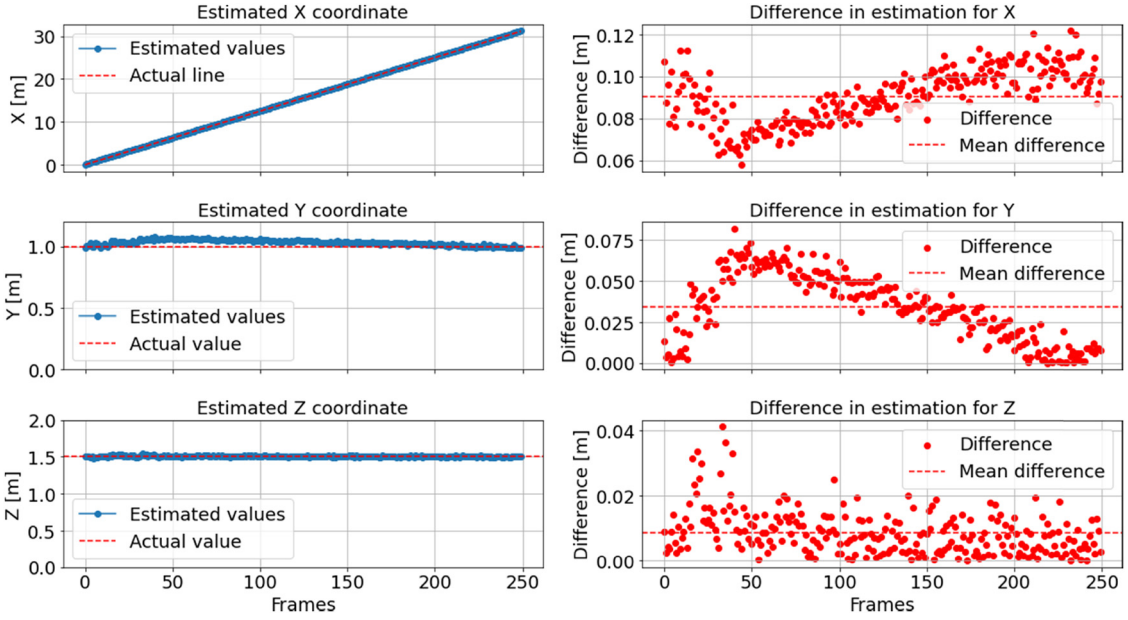

Figure 8. Comparison of estimated and actual coordinates for the tracked model mass based on screenshots triangulation, along with the corresponding estimation differences.

Source: Author's contribution.

unequivocally verify the accuracy of the point's location over time. The localization results of the mass modelled from Figure 6 are presented in Figure 8.

The histograms presented in Figure 9 illustrate the distribution of estimation errors for the *X*, *Y*, and *Z* coordinates of the mass position. The errors were calculated as absolute differences between the estimated and actual positions for each frame.

The chart includes a red dashed line indicating the 95th percentile of the error values, along with a summary box showing key statistical metrics: standard deviation, mean squared error (MSE), root mean squared error (RMSE), and maximum error. This visualization provides a clear overview of the accuracy and consistency of the estimation process. Based on the obtained results and error distribution charts, it can be concluded that the system achieved high accuracy in estimating the vehicle's axle position. These results confirm that vehicle tracking and localization on the structure using camera-based systems supported by ML can be performed with high precision.

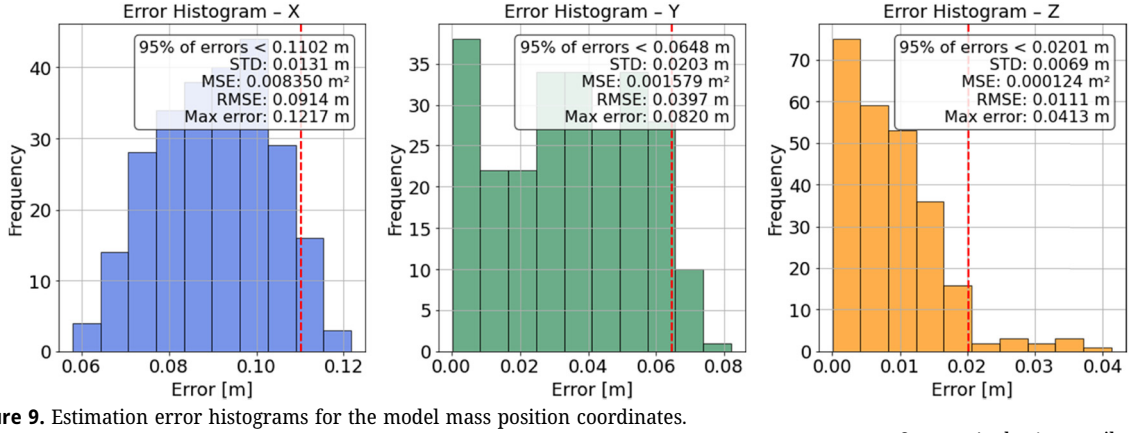

Figure 9. Estimation error histograms for the model mass position coordinates.

Source: Author's contribution.

4. ML tools in LIS

4.1 Application of autoencoders

Determining vehicle mass and axle pressures is a particularly challenging task. A neural network, using only the response signals of bridge structure and information about the vehicle's configuration and position over time, must reduce the data to key information and based on this, estimate axle pressures. Additionally, the bridge structure's response is sensitive to measurement noise and structural dynamics.

Classifying this task type, it is undoubtedly a case of time series analysis and dimensionality reduction. Network types specializing in such tasks are autoencoders and noise reduction techniques (Table 2).

Autoencoders are a class of artificial neural networks that specialize in generating dense data representations, known as encodings or hidden representations, without external supervision. These encodings have reduced dimensionality compared to the input data, making them ideal for dimensionality reduction applications, especially in the context of data visualization. Autoencoders are also used as feature detectors and tools for unsupervised pretraining in the context of deep neural networks [49].

The mechanism of autoencoders involves reconstructing input signals at the network outputs, which may seem like a simple task. However, the introduction of constraints, such as reducing the size of hidden representations or adding noise to the output data, significantly complicates the process.

Structurally, an autoencoder is divided into two main parts: an encoder, which converts input data into a hidden representation, and a decoder, which regenerates the output data from this hidden representation. An example of autoencoder's operation is presented in Figure 10. For a 2D image, such as a painting, the autoencoder analyses each pixel and searches for patterns that are essential for reconstructing the image. Instead of storing complete information about each pixel, it reduces the data to key elements, such as shapes, contours, or characteristic arrangements of light and dark areas. These simplified features are then used to recreate the image. While the reconstructed image will not be a perfect replica of the original, it will retain its most important characteristics, such as recognizable contours of the objects it depicts. In the case of a 1D signal, such as waveform representing measurement data, the autoencoder analyses the signal's progression over time. It extracts key information, such as repeating patterns, amplitudes, or frequencies, which define the signal's character. Using these features, the signal is reconstructed. Although the reconstruction may not perfectly match the original, the essential characteristics of the signal, such as its shape or dominant frequencies, remain intact. Autoencoders are particularly useful when the goal is to focus on the essence of the data while ignoring redundant details. For images, they enable data compression; for signals, they allow for cleaning a noise or analysing key features. Their ability to extract information and reconstruct data makes them valuable in various applications, from computer vision to processing audio and biological signals.

According to the literature [1], various basic types of autoencoders can be distinguished:

- **Stacked autoencoder** – consists of multiple hidden layers that successively compress the information contained in the input data into a smaller representation. Each layer learns increasingly abstract features, enabling the network to capture complex patterns. As a result, it allows for more accurate data reconstruction or more effective extraction of key information from large datasets.
- **Denoising autoencoder** – works by introducing controlled noise into the input data during training, forcing the network to ignore disturbances and focus on the essential features. This approach teaches the model not only to reconstruct the data but also to effectively denoise it, which is particularly useful when the data are affected by noise.
- **Convolutional autoencoder** – based on convolutional layers and subsampling (pooling) operations, it effectively captures local patterns and hierarchical structures in spatial data such as images. It is especially useful for image compression and reconstruction, as well as for extracting important visual features while maintaining a low computational cost.
- **Recurrent autoencoder** – uses recurrent neural connections, enabling the analysis of data with a sequential or temporal nature. This makes it well-suited for modelling dynamic data such as time series, biological signals, or text sequences.
- **Sparse autoencoder** – introduces additional constraints to limit the number of simultaneously active neurons in hidden layers, resulting in a so-called "sparse" data representation. This enables efficient extraction of only the most relevant features needed for reconstruction, reducing redundancy and lowering the risk of overfitting.
- **Variational autoencoder (VAE)** – based on a probabilistic approach, it learns to represent data as a probability distribution in the latent space. This enables

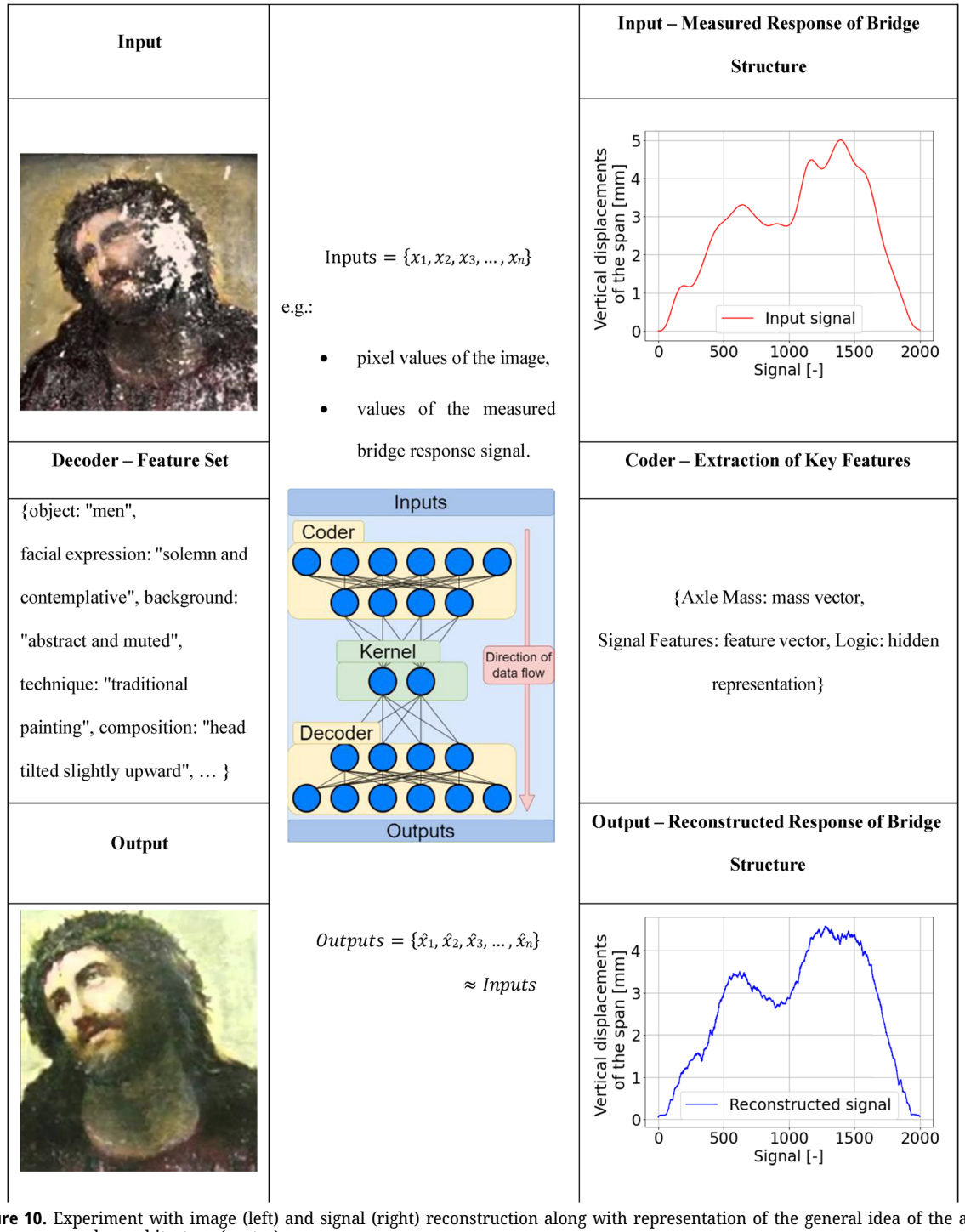

Figure 10. Experiment with image (left) and signal (right) reconstruction along with representation of the general idea of the auto-encoder architecture (centre).

Source: Author's contribution.

not only compression and reconstruction of data, but also the generation of new examples through random sampling from the learned distribution. VAEs are especially valuable in generative ML applications, such as generating realistic images or simulating complex signals.

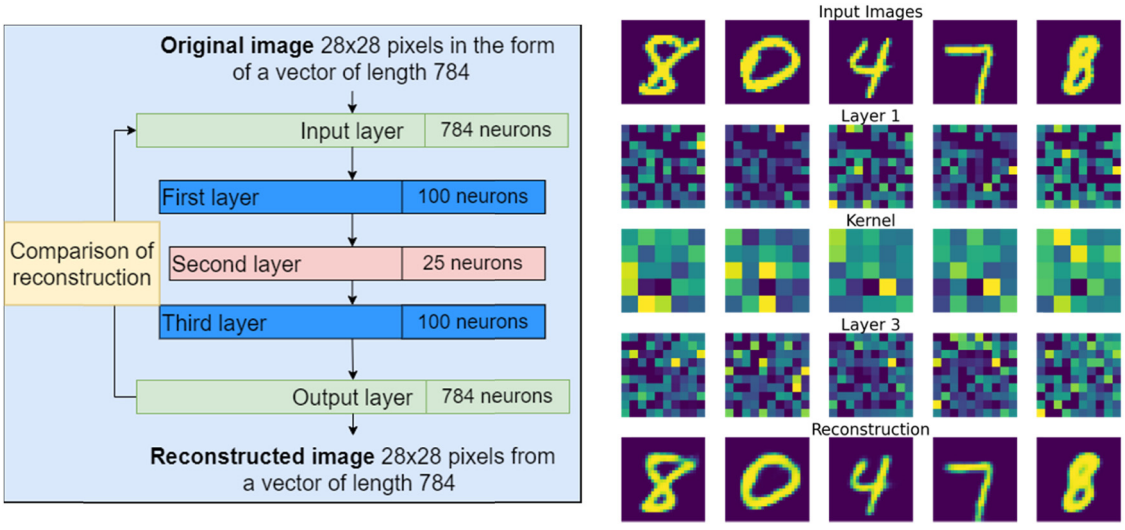

Figure 11. (a) Stacked autoencoder architecture and (b) results of the network operation in following steps.
Source: Author's contribution.

It is noted that networks can have a complex structure and simultaneously represent multiple classes. The example in Section 4.2 uses a stacked denoising autoencoder based on convolutional layers. In the following part of this subsection, the operation mechanism of selected autoencoders is described and presented.

An example of a stacked autoencoder implementation is presented, with the task of learning to classify digits, extract characteristic information, and then reconstruct the input image based on a set of handwritten images. To demonstrate the mechanisms of network operation, the MNIST database [50] was used, which is an extensive database of handwritten digits and is commonly used to train various image processing systems. The MNIST database contains 60,000 images for training and 10,000 images for testing. Selected input images, the activations of individual neurons in the encoder and decoder, and the reconstructed input images are presented in Figure 11.

The implementation of the stacked autoencoder, shown in Figure 11a, includes two main components: an encoder and a decoder. Encoding begins by transforming input data, such as 28 × 28 pixel monochromatic images, into a 784-element vector. These vectors then pass through layers with a decreasing number of units, allowing for the extraction and compression of key features. The decoder transforms the encodings through layers with an increasing number of units, leading to the reconstruction of the original images.

The activations of individual neurons in each layer are presented in Figure 11b for five randomly selected images of digits. It is noticeable that the network can reduce the values representing the digit to just 25 values and then reconstruct the image while preserving distinctive features such as handwriting style and imperfections in pen pressure on the paper.

Another capability of autoencoders is their ability to denoise images with visual interference. This capability can be achieved by adding a dropout layer, which disables the weakest connections between neurons in the neural network. It is worth mentioning here that neural networks operate according to Hebb's rule [51], which postulates that when one biological neuron regularly stimulates another neuron, their synaptic connection strengthens. By removing the weakest connections, we reduce the computational complexity of the neural network, speeding up its learning and operation process, and additionally make the network less sensitive to irrelevant information. The Dropout layer accepts a threshold value for connections between neurons, below which those connections are disabled.

Figure 12 shows the structure of an autoencoder network with a denoising function, which also includes an encoder and a decoder. Encoding begins by transforming input data, such as 28 × 28 pixel monochromatic images, into a 784-element vector. These vectors then pass through layers with a decreasing number of units, allowing for the extraction and compression of key features. Additionally, a dropout layer is added, which reduces the number of connections between neurons by disabling the weakest connections.

The decoder transforms the encodings through layers with an increasing number of units, leading to the reconstruction of the original images.

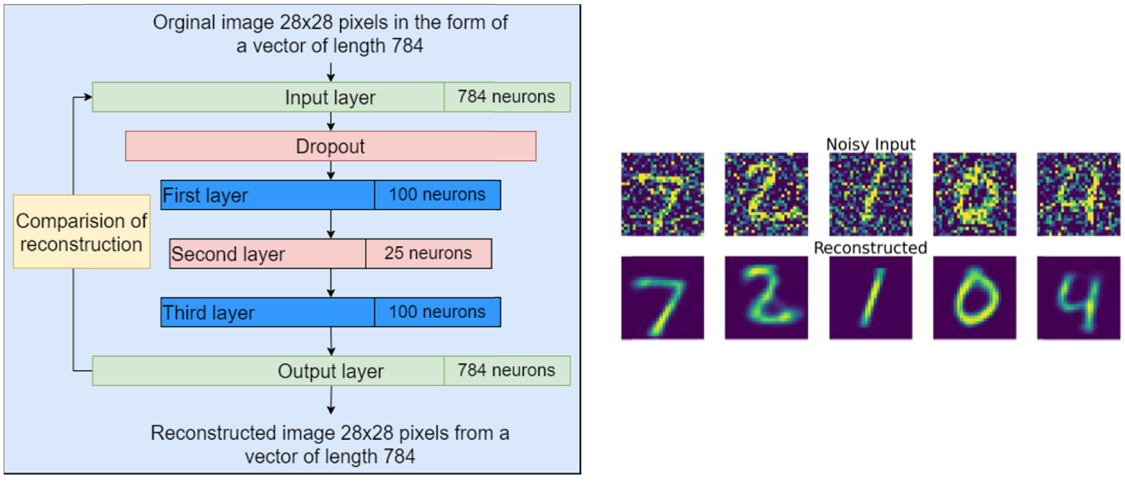

Figure 12. (a) Denoising autoencoder architecture and (b) results of the network operation in following steps.
Source: Author's contribution.

It is noticeable that the network can interpret a heavily noisy image, extract the most important features, and clean the image.

Another tool in image analysis comprises convolutional autoencoders, which are key elements in pattern recognition, especially in image processing. Their unique architecture, based on convolution and subsampling operations, enables effective feature extraction and dimensionality reduction of input data [52,53]. The convolution operation, which is the basis of CNNs, involves moving a filter (convolution kernel) across the entire input image, generating a feature map. This allows the network to focus on local features of the image, enabling edge, shape, or texture detection in the initial layers, and gradually composing more complex patterns in deeper layers. These learnable filters automatically adapt during training, detecting specific features. Convolutional filters, which play a key role in feature extraction, can be seen as small, learnable kernels focused on detecting specific patterns in the input data. Subsampling, often using max or average pooling, reduces the size of the data while preserving essential information. As data pass through alternating convolution and pooling layers, the network progressively extracts more abstract features, enabling deeper understanding of image content. In convolutional autoencoders, this process is used not only for feature extraction but also for reconstructing the input image from a compressed representation. Figure 13 illustrates a typical CNN structure, showing the gradual abstraction from low-level features to high-level representations. In CNNs, these layers are arranged alternately. Additionally, many filters and layers are used, allowing the network to capture more complex features.

Despite the use of multiple filters and operations, CNN-based autoencoders typically require fewer parameters. Their architecture is also highly efficient, especially when accelerated by graphics processing units (GPUs) optimized for matrix operations. In the case of an

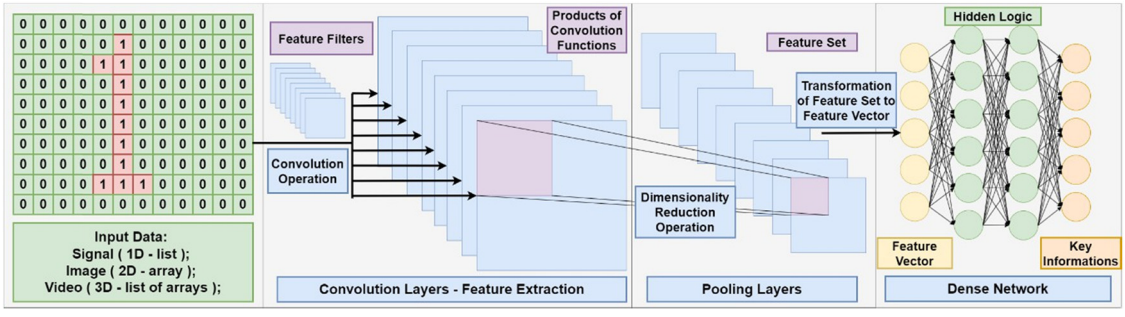

Figure 13. Typical structure of a CNN.
Source: Author's contribution.

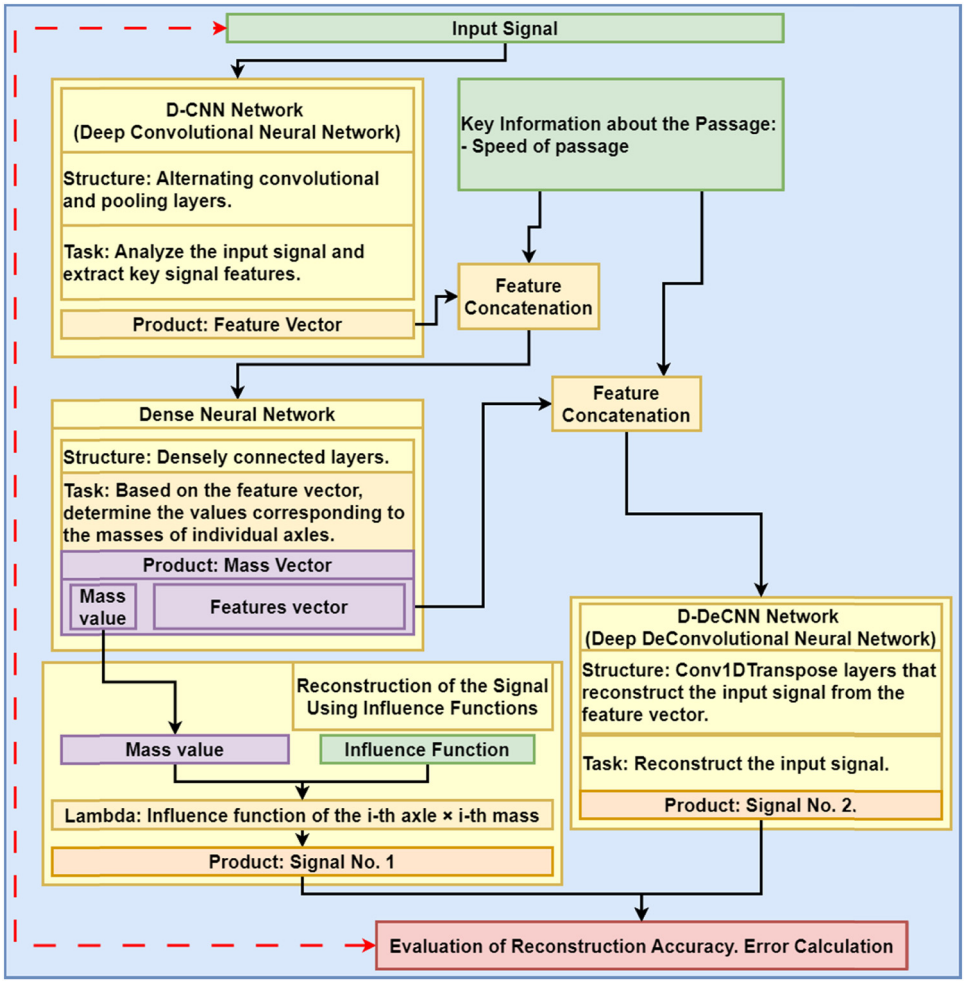

Figure 14. General block diagram of the proposed system algorithm and operation.

Source: Author's contribution.

autoencoder based on convolutional networks, we can leverage this mechanism by using deep layers to deconstruct the image into a feature vector and then reconstruct the original image based on this vector. Despite the apparent complexity and the need to design multiple filters and perform numerous operations of multiplying images by filters, this network is characterized by a smaller number of parameters.

4.2 Example of an LIS based on autoencoders

This section presents an algorithm designed to determine the load exerted by a sprung mass (as a model of vehicle) moving across a bridge based on a response of the structure.

The proposed algorithm is inspired by the Moses algorithm [22], but it incorporates ML techniques, including the autoencoders described above. The primary objective is to demonstrate that ML can implicitly learn to solve the problem without the need to implement traditional algorithms or equations.

There are several assumptions for the proposed system:

1. The algorithm must be self-adaptive by employing an autoencoder algorithm based on unsupervised learning, where the input signal is also used as the expected output. This approach enables the model to learn a compact internal representation of the structural response and continuously improve without the need for labelled data.
2. The bridge response signal is a discrete signal represented as a vector of information, which reflects the

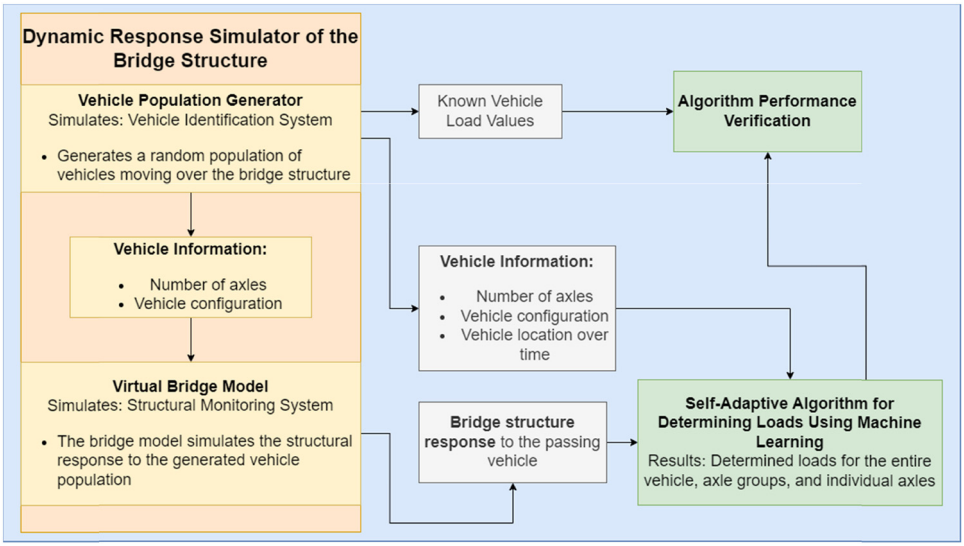

Figure 15. Implementation and testing scheme using mock system components.

Source: Author's contribution.

vertical displacement of the structure at midspan during the vehicle's passage.

3. The applied VIS has already been successfully implemented on the structure and provides accurate information regarding the mass position coordinates on the bridge during passage and speed.
4. The bridge structure supports a single sprung mass modelling a vehicle.
5. IF of the measured bridge response is known.

Figure 14 illustrates the architecture of the proposed system.

The input parameters for the system (given in green in Figure 14) include the following:

- **Input signal**: A discrete vector representing the measured response of the bridge structure. The first value describes the bridge structure's response when the mass appears on the bridge, and the last value represents the moment when the moving mass leaves the structure. A constant time step between each value is assumed.
- **IF**: A function representing the influence of the measured bridge response, depending on the changing position of the vehicle as a function of time.
- **Mass location over time**: A function describing the mass position on the bridge as the mass moves.

The output parameters of the system include the following:

- **Output signals:** The reconstructed signals of the dynamic response of the bridge structure.
  - **Signal No. 1:** Quasi-static structural response. The response obtained using the IF of the measured value.
  - **Signal No. 2:** Dynamic structural response. The response obtained based on the vehicle's mass and the feature vector.

The network is characterized by a linear structure in which the input information is transmitted through the network in one direction, without branching, until it reaches the output layer to determine the function reconstruction error.

The input to the block is the bridge structure response signal. The task of the CNN layers is to extract features using filters and convolution functions. Since the analysed signal is a vector, 1D convolutional layers are used. Information about the mass transit speed is added to the obtained feature vector. This prepared feature vector is then combined and analysed by the dense layers of the neural network. The value determined by the neuron is multiplied by the influence functions in a lambda function. The products are then summed, resulting in the output signal.

Signal no. 2 is obtained based on the estimated moving mass, the feature vector, and information about the passage. Using inverse convolutional layers, the neural network reconstructs the structural response signal from the feature vector. Since this method does not use IFs, the reconstructed signal will account for both the static and dynamic nature of the bridge structure's response.

The program compares the differences between the input and output signals by calculating the MSE. Since the signals are in discrete form, the difference between the expected (input) value and the actual value can be determined for each signal point. The optimization

algorithm then attempts to minimize the error, for example, through backpropagation.

MSE is a popular function used in regression, measuring the average squared differences between predicted and actual values. It is expressed by the formula

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2,$$

where Y_i represents the actual values, $\hat{Y}_i$ represents the values predicted by the model, and n is the number of samples.

In our study, we adopt an incremental learning approach, which is particularly suitable for B-WIM systems operating in real-world conditions. Unlike batch learning, where the model is trained once on a fixed dataset, incremental learning allows the model to adapt continuously using data being collected during operation, in this case, from each passing vehicle. The model used is based on autoencoders and is continuously updated as new data are gathered from each passing vehicle.

The adaptation mechanism is guided by the minimization of the MSE, using the Adam optimizer [54], which combines the advantages of both Adaptive Gradient Algorithm, and Root Mean Squared Propagation for efficient and robust training. A regularization strategy based on an early stopping criterion is employed to prevent overfitting to short-term fluctuations while maintaining training efficiency. This ensures that the model adapts without overfitting to short-term fluctuations, while maintaining training efficiency.

The presented algorithm is an example of unsupervised learning, meaning it does not require labelled data. The input data, which is the bridge structure response signal, also serve as the target expected outcome. This leads to the initialization of the neural network on the tested object, and over time, as the history of vehicle transits grows, the neural network has an increasing amount of training data.

The proposed model is a moderately deep convolutional autoencoder implemented using Keras, TensorFlow, and Python. It consists of a series of 1D convolutional and transposed convolutional layers, followed by dense layers for load prediction and signal reconstruction. The total number of trainable parameters is approximately 126,000.

The network was trained and tested on a budget-level consumer GPU (NVIDIA GeForce GTX 1660), demonstrating that the training process is feasible even without access to high-performance computing infrastructure. In the current configuration, training or updating the model with new data (i.e. incremental learning) requires only a few seconds per batch, which makes it suitable for periodic or event-triggered retraining in real-time or near-real-time applications.

Given the low model complexity, the use of early stopping to limit redundant training cycles, and the modular structure of the system, the computational cost is minimal. This makes it possible to deploy and update the model on embedded systems, such as industrial PCs or edge AI platforms, within the constraints of typical B-WIM setups.

To test the system concept, the technique of testing the system using mock objects was employed and presented in Figure 15. Mock objects play a key role in the software development process, especially in unit testing. They enable the simulation of behaviours and interactions of real objects in an isolated environment, allowing for precise and controlled testing of software components.

The goal of testing is to obtain data that corresponds to the information that would be collected from a real system. A dynamic response simulator of the bridge structure is used, consisting of two main mock components:

- **Vehicle population generator** – simulates results of the VIS
- **Virtual bridge model** – replaces the structural monitoring system and the bridge response to a passing vehicle.

Since the discussed example serves as a proof of concept, a mathematical model of a bridge is used to address the problem of a sprung mass moving across a simply supported beam (Figure 16).

The bridge model is based on the classical beam theory according to Euler-Bernoulli. The following assumptions are adopted:

- The beam is made of a linear elastic material; basic characteristics, like Young's modulus (E) and the

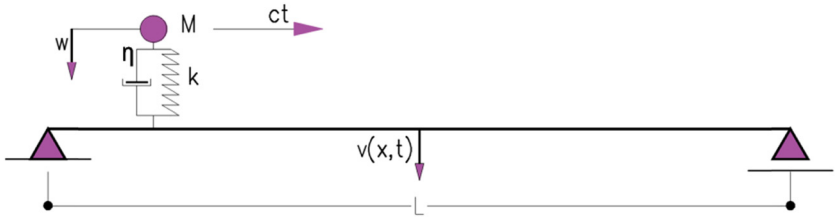

Figure 16. Virtual bridge model.

Source: Author's contribution.

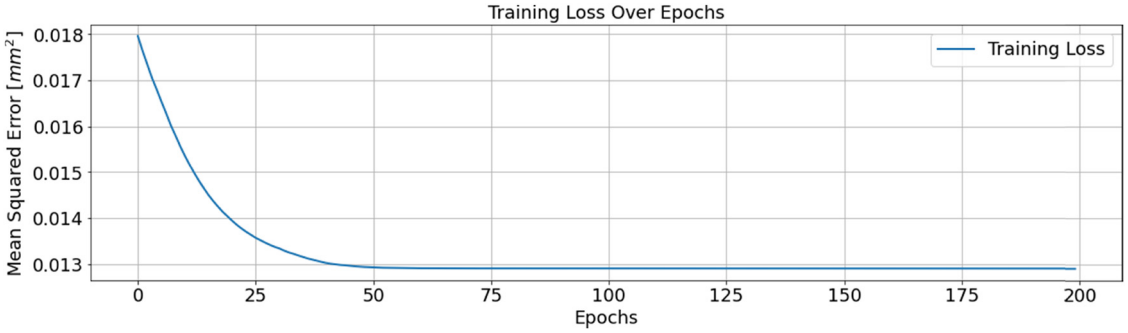

Figure 17. Learning curve values for training data.

Source: Author's contribution.

moment of inertia of the cross-section (I), are constant throughout the entire element.
- The model does not account for the effects of friction, thermal interactions, or other nonlinearities.
- A sprung mass moves over the structure at a constant velocity, connected to the base by a damper and a linear spring.

According to the literature, the solution for a typical Euler beam can be expressed using modal superposition. Ultimately, equation for the *j*th vibration mode of the beam is presented in (1), and the equation for the vibrations of the sprung mass is provided in (2).

$$V_j(t) + 2\omega_b V_j(t) + \omega_j^2 V_j(t) = \frac{2}{\mu l}\left(\mathrm{Mg} - M\frac{\mathrm{d}^2 w(t)}{\mathrm{d}t^2}\right)\sin\frac{j\pi ct}{L} \quad (1)$$
$$(j = 1,\ 2,\ \ldots,\ \infty),$$

$$M\frac{\mathrm{d}^2 w(t)}{\mathrm{d}t^2} + \eta\frac{\mathrm{d}w(t)}{\mathrm{d}t} + kw(t) = k[w(t) - v(x = ct,\ t)], \quad (2)$$

where $v(x, t) = \sum_{i=1}^{\infty} \sin\frac{i\pi x}{L} V_i(t) v(x, t)$ is the vertical displacement of the beam at point x at time t, $V_i(t)$ is the modal displacement, μ is the linear mass of the beam, L is the span length of the beam, c is the velocity of the moving sprung mass, M is the moving mass, g is the gravitational acceleration, k is the suspension stiffness, and η is the damping coefficient of the mass suspension.

The presented problem was solved in Python using the *scipy.odeint* package for solving differential equations.

Subsequently, a population of sprung masses consisting of 10,000 masses was generated. Parameters such as mass, suspension stiffness, damping, and travel speed were randomly selected, subject to the constraints of the specified boundary conditions:
- The mass M_i ranges between 1 and 50 tons.
- The damping is expressed as a fraction of the critical damping $c_{kr} = 2\sqrt{\mathrm{km}}$ and lies within the range 0–0.5 c_{kr}.
- The suspension stiffness was randomly selected such that the spring displacement due to the gravitational force of the mass, $\delta = \frac{\mathrm{Mg}}{k}$, falls within the range of 0.1–0.5 m.
- The velocity is randomly selected within the range of 1–30 m/s.

The results of the system verification are demonstrated by means of learning curve charts shown in Figure 17 presenting the change in MSE during the neural network training process. It shows how the model learns to solve the task during training, allowing for the assessment of model convergence and identification of potential issues, such as overfitting or underfitting.

Three randomly selected results of the elaborated LIS system operation are given in Table 4 presenting a comparison of calculated mass by the model with the original mass obtained from the simulator at different stages of training.

To evaluate the accuracy of the LIS system, Figure 18 presents a comparison of the reconstructed signals generated by the model with the original signal obtained from the simulator at various stages of training.

Both the results presented in Table 4 and the graphs shown in Figure 18 confirm that the dynamic response of the model accurately reproduces the dynamic behaviour of the structure using only limited input data, such as the estimated mass and

	Static loads expressed as mass (*t*)		
	Mass no. 1	Mass no. 2	Mass no. 3
Actual value	34.4	28.11	50.23
Before training	0.00	0.06	0.20
After training	34.6	28.23	49.99
Error	0.57%	0.41	0.48

Table 4. Determined static load values at various stages of network training.

Source: Author's contribution.

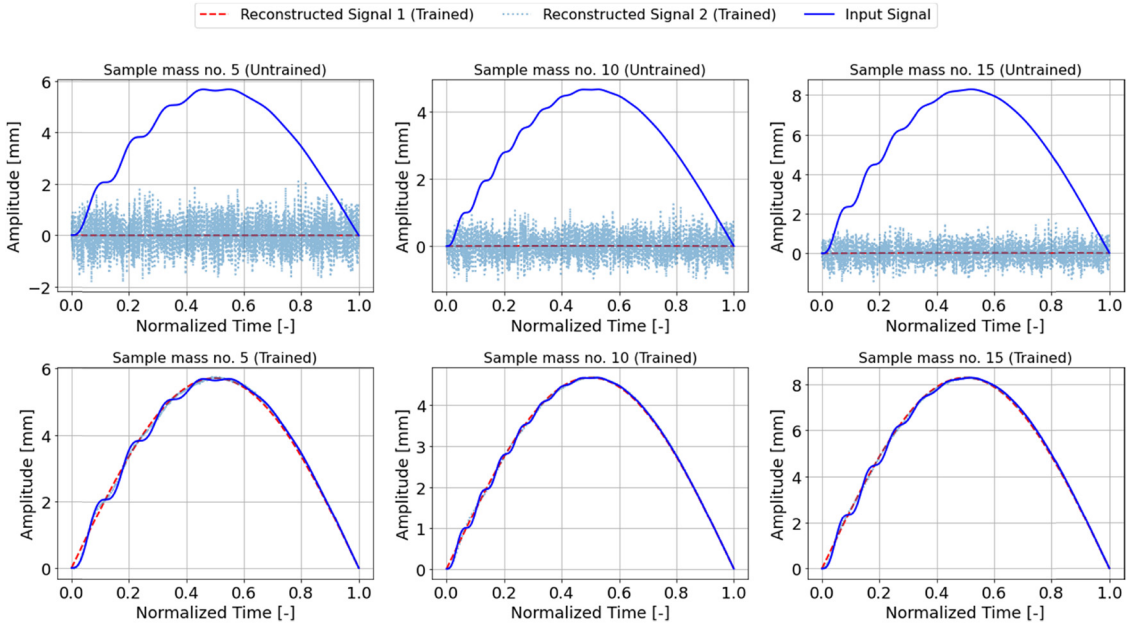

Figure 18. Comparison of reconstructed signals with the original for different training phases.

Source: Author's contribution.

feature vector. The system leveraged information from tens of thousands of transits to uncover the logic behind moving mass dynamic behaviour and to model it effectively.

To evaluate the accuracy of the system, a new population of 2,000 masses was generated. These masses were not presented to the neural network during the training process. The

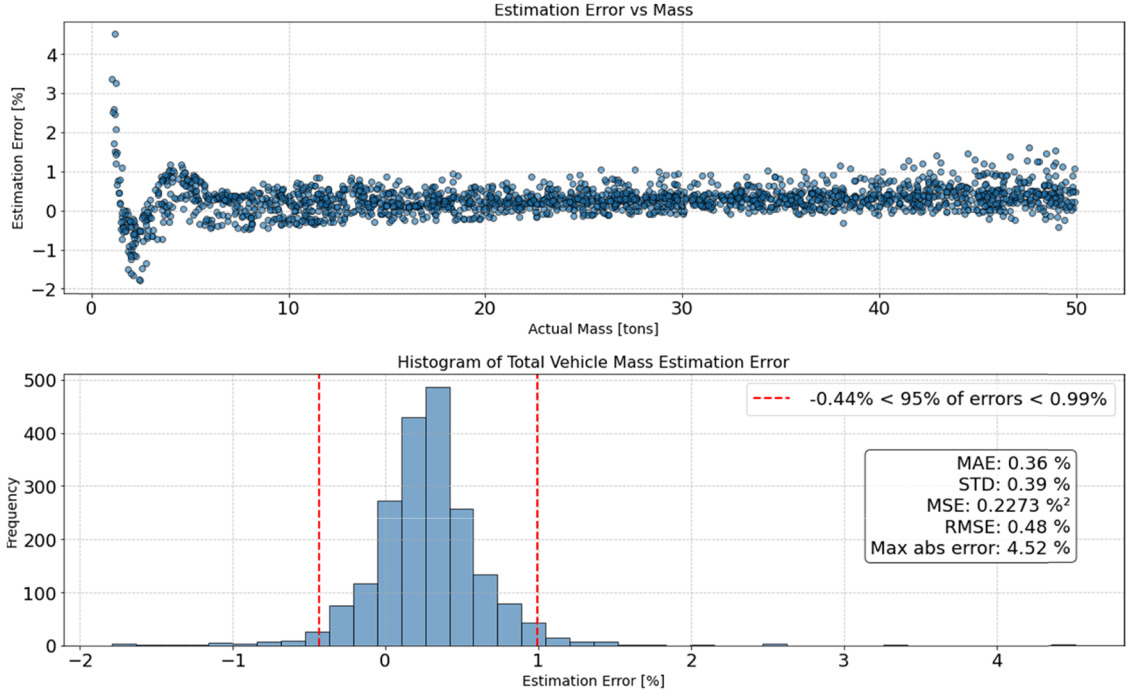

Figure 19. The estimation error for the test population.

Source: Author's contribution.

estimation error obtained for this population dependent on the considered mass value is presented in Figure 19.

The estimation error for the test population falls within reasonable limits. Based on the results, the estimation error remains below −0.44 and 0.99% for 95% of the tested samples. The mean absolute error is approximately 0.36%, and the RMSE is 0.48%, indicating high consistency in the predictions. These outcomes demonstrate that the model is capable of accurately estimating vehicle mass across a wide range of values, which is critical for real-time vehicle classification and monitoring applications.

It is worth emphasizing that the algorithm is able to determine accurately the travelling mass without the need for creating complex computational models or intricate formulas.

5. Summary

5.1 Summary of key findings

The conducted research confirms feasibility of applying ML methods to estimate the weight of vehicles crossing bridge structures. B-WIM systems require the integration of two fundamental subsystems: the VIS and the LIS. The roles and basic classification of these components are presented and discussed, and the results demonstrate that ML methods effectively support both subsystems.

The first key task addressed in the study is vehicle classification and localization. Using a simulation model, the study shows that a vision-based system employing the YOLO algorithm is capable of tracking and classifying vehicles in motion. The system achieves high accuracy: 95% of localization errors remain below 11 cm over a 30 m tracking segment. These results validate the core concept and indicate the potential for further development.

The second task focuses on estimating vehicle's mass based on the structural response. The study demonstrates that this inverse problem can be successfully solved using ML techniques. A particularly noteworthy aspect is the use of unsupervised learning through autoencoders, which enable the system to improve its accuracy over time without requiring labelled training data or human intervention in the learning process. The autoencoder is trained to reconstruct its own input, allowing it to capture the essential structure of the signal and detect deviations in future cases. However, it is important to recognize the limitations of the proposed solution. The neural network relies on an IF, which establishes a physical link between the structural response and the applied force. This function may be determined during the calibration phase using a reference vehicle. As such, it does not account for long-term structural changes (e.g. due to material degradation, damage, or rheological effects).

The application of ML methods in B-WIM-type systems can not only improve load identification accuracy but also enhances infrastructure resilience and durability by better adapting to real-world conditions. In the context of maintaining the proper condition of civil infrastructure, it is essential to develop effective systems for monitoring operational loads. These systems should enable the following:

- Managing the traffic to limit maximum loads on bridge structures in cases of detected accumulation of overloaded vehicles. Through electronic road signs, it would be possible to temporarily close specific lanes on bridge structures and introduce speed limits to reduce damage to both the road surface and the bridge structure.
- Updating and monitoring the changes and trends in traffic patterns to optimize the design of new road sections and the modernization of existing ones. This could include adjusting the requirements for monitored roads and bridge structures based on the determined load levels, through increasing or decreasing structural demands.
- Monitoring loads in the context of fatigue processes in structures, providing a reliable basis for determining the future usability of many bridge structures.
- Updating load models in design standards for road and bridge structures to reflect contemporary traffic conditions.

Despite existing challenges, such as the need for large training datasets and proper system calibration, the development of this technology indicates promising possibilities for further automation and optimization of bridge and road monitoring processes.

5.2 Future works and other potential applications

The further research in the presented area is aimed at extending the proposed B-WIM system based on ML for identification of multi-axle vehicles. Furthermore, addressing real-time requirements in future B-WIM systems will require deeper integration with vision-based systems, Internet of Things sensor networks, and edge computing infrastructure to enable continuous, low-latency data acquisition and processing. Long-term challenges, such as model drift and progressive structural degradation, also need to be considered to ensure sustainable system

performance. These important aspects may be explored in future studies.

A particularly promising research direction is the design of systems that do not rely exclusively on predefined IFs, or that can update them automatically. This could be achieved through data-driven identification methods, integration with structural health monitoring systems, or hybrid physical-data models capable of learning changes in the mechanical behaviour of the structure.

Funding information

Authors state no funding involved.

Author contributions

AM – Conceptualization; Methodology; Software; Validation; Formal analysis; Investigation; Data curation; Visualization; Project administration; Writing – original draft; Writing – review & editing. TK – Conceptualization; Supervision; Writing – review & editing. JB – Conceptualization; Supervision; Writing – review & editing.

Conflict of interest statement

Authors state no conflict of interest.

References

[1] Géron, A. (2020). *Handson machine learning with Scikit-learn, Keras & tensorflow* (3rd ed.) Helion.

[2] Chrystal, R. China, 'Five machine learning types to know', IBM Think. [Online]. https://www.ibm.com/think/topics/machine-learning-types.

[3] Lowe, D.J., Emsley, M.W., Harding, A. (2006). Predicting construction cost using multiple regression techniques. *Journal of Construction Engineering and Management*, *132*(7), 750–758. doi: 10.1061/(ASCE)0733-9364(2006)132:7(750).

[4] Thomas, N., Thomas, A.V. (Nov. 2016). Regression modelling for prediction of construction cost and duration. *AMM*, *857*, 195–199. doi: 10.4028/www.scientific.net/AMM.857.195.

[5] Sheshadri, A., Marathe, S., Rodrigues, A.P., Nieświec, M. (Mar. 2025). Predictive modelling of pavement quality fibre-reinforced alkali-activated nano-concrete mixes through artificial intelligence. *Studia Geotechnica et Mechanica*, *46*(s1), 389–416. doi: 10.2478/sgem-2025-0007.

[6] Fang, Q., Ibarra-Castanedo, C., Garrido, I., Duan, Y., Maldague, X. (May 2023). Automatic detection and identification of defects by deep learning algorithms from pulsed thermography data. *Sensors*, *23*(9), 4444. doi: 10.3390/s23094444.

[7] Ji, H., Hwang, I., Kim, J., Lee, S., Lee, W. (Dec. 2024). Leveraging feature extraction and risk-based clustering for advanced fault diagnosis in equipment. *PLoS ONE*, *19*(12), e0314931. doi: 10.1371/journal.pone.0314931.

[8] Liebert, A., Weber, W., Reif, S., Zimmering, B., Niggemann, O. (2022). Anomaly detection with autoencoders as a tool for detecting sensor malfunctions. In *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)* (pp. 1–8). doi: 10.1109/ICPS51978.2022.9816908.

[9] Ghanbari, M., Hafizi, K., Bano, M., Ebrahimi, A., Hosseinzadeh, N. (2023). *GPR data processing framework for detecting subsurface utilities*. doi: 10.13140/RG.2.2.13263.15528.

[10] Li, Y., Dang, P., Xu, X., Lei, J. (Mar. 2023). Deep learning for improved subsurface imaging: Enhancing GPR clutter removal performance using contextual feature fusion and enhanced spatial attention. *Remote Sensing*, *15*(7), 1729. doi: 10.3390/rs15071729.

[11] Stergiou, K., Ntakolia, C., Varytis, P., Koumoulos, E., Karlsson, P., Moustakidis, S. (Mar. 2023). Enhancing property prediction and process optimization in building materials through machine learning: A review. *Computational Materials Science*, *220*, 112031. doi: 10.1016/j.commatsci.2023.112031.

[12] Jocz, M., Lefik, M. (Dec. 2023). Correlation between Cone Penetration Test parameters, soil type, and soil liquidity index using long short-term memory neural network. *Studia Geotechnica et Mechanica*, *45*(s1), 405–415. doi: 10.2478/sgem-2023-0023.

[13] Rolf, B., Jackson, I., Müller, M., Lang, S., Reggelin, T., Ivanov, D. (Oct. 2023). A review on reinforcement learning algorithms and applications in supply chain management. *International Journal of Production Research*, *61*(20), 7151–7179. doi: 10.1080/00207543.2022.2140221.

[14] Farrokhi, F., Rahimi, S. (Sep. 2020). Supervised probabilistic failure prediction of tuned mass damper-equipped high steel frames using machine learning methods. *Studia Geotechnica et Mechanica*, *42*(3), 179–190. doi: 10.2478/sgem-2019-0043.

[15] Rys, D. (2015). Heavy vehicle road loads and their impact on the fatigue durability of flexible and semi-rigid pavement structures. *Politechnika Gdańska Wydział Inżynierii Lądowej i Środowiska, Gdańsk* (p. 201). https://pbc.gda.pl/Content/50418/phd_ry%C5%9B_dawid.pdf.

[16] Ryś, D., Judycki, J., Jaskuła, P. (2014). The impact of overloaded vehicles on the durability of asphalt pavements. *Logistyka*, *6*, 9318–9328.

[17] Normann, O.K., Hopkins, R.C. (1952). Weighing vehicles in motion. *Highway Research Board*, *50*(221), 29.

[18] Burnos, P. (2014). Weighing road vehicles in motion. Part 1: The impact of overloaded vehicles on pavement. *Drogownictwo*, *6/2014*.

[19] Burnos, P. (2014). Weighing Road Vehicles in Motion. Part 2: Types and Characteristics of Weigh-In-Motion (WIM) systems. *Drogownictwo*, *6/2014*.

[20] Burnos, P. (2014). Weighing road vehicles in motion. Part 3: Pressure sensors used in Weigh-In-Motion (WIM) systems. *Drogownictwo*, *6/2014*.

[21] Burnos, P. (2014). Weighing road vehicles in motion. Part 4: Accuracy assessment of Weigh-In-Motion (WIM) systems. *Drogownictwo*, *6/2014*.

[22] Moses, F. (Jan. 1979). Weigh-In-Motion system using instrumented bridges. *Transportation Engineering Journal of ASCE*, *105*(3), 233–249. doi: 10.1061/TPEJAN.0000783.

[23] Cardini, A.J., DeWolf, J.T. (Nov. 2009). Implementation of a long-term bridge Weigh-In-Motion system for a steel girder bridge in the interstate highway system. *Journal of Bridge Engineering*, *14*(6), 418–423. doi: 10.1061/(ASCE)1084-0702(2009)14:6(418).

[24] Quilligan, M. (2003). Bridge weigh-in motion: Development of a 2-D multi-vehicle algorithm. *Trita- BKN. Bulletin*, *69*, 144.

[25] O'Brien, E., Žnidarič, A., Eds. (2001). *Weighing-in-motion of axles and vehicles for Europe (WAVE). Report of work package 1.2: Bridge WIM systems (B-WIM)*. Ljubljana: Zavod za gradbeništvo Slovenije.

[26] Kalin, J., Žnidari, A. (2019). *Practical implementation of Nothing-On-the-Road Bridge Weigh-In-Motion system*. https://hvttforum.org/wp- content/uploads/2019/11/Practical-Implementation-of-Nothing-on-the-Road-Bridge-Weigh-In-Motion-System-Kalin.pdf.

[27] Peters, R. (Aug. 1984). AXWAY - a system to obtain vehicle axle weights. *Presented at the 12th ARRB Conference, Hobart, Hobart, TAS, Australia*. Vermont South, VIC, Australia: ARRB Group Limited.

[28] Jacob, B., Brien, E.J.O., Jehaes, S. (2002). Weigh-in-motion of road vehicles. Final report of the COST 323 Action (WIM-LOAD). *Laboratoire Central des Ponts et Chausse´es, Paris*. https://www.is-wim.org/doc/wim_eu_specs_cost323.pdf.

[29] Li, C.Y., Wang, C., Yang, Q.X., Qi, T.Y. (Sep. 2022). Identification of vehicle loads on an orthotropic deck steel box beam bridge based on optimal combined strain influence lines. *Applied Sciences*, *12*(19), 9848. doi: 10.3390/app12199848.

[30] Chatterjee, P., O'Brien, E.J., Li, Y., González, A. (Nov. 2006). Wavelet domain analysis for identification of vehicle axles from bridge measurements. *Computers & Structures*, *84*(28), 1792–1801. doi: 10.1016/j.compstruc.2006.04.013.

[31] Yu, Y., Cai, C.S., Cai, C.S., Cai, C.S., Deng, L. (Oct. 2017). Vehicle axle identification using wavelet analysis of bridge global responses. *Journal of Vibration and Control*, *23*(17), 1077546315623147. doi: 10.1177/1077546315623147.

[32] Dunne, D., O'Brien, E.J., Basu, B., Gonzalez, A. (Jan. 2005). Bridge WIM systems with Nothing On the Road (NOR). In *Proceedings of the 4th International WIM Conference* (pp. 109–117). Zurich: International Society for Weigh-in-Motion.

[33] Machelski, C., Hildebrand, M. (Feb. 2015). Efficiency of monitoring system of a cable-stayed bridge for investigation of live loads and pier settlements. *Journal of Civil Structural Health Monitoring*, *5*(1), 1–9. doi: 10.1007/s13349-014-0074-7.

[34] Chan, T.H.T., Law, S.S., Yung, T.H. (Oct. 2000). Moving force identification using an existing prestressed concrete bridge. *Engineering Structures*, *22*(10), 1261–1270. doi: 10.1016/s0141-0296(99)00084-x.

[35] Zhu, X., Law, S.S., Law, S.S. (Sep. 2000). Identification of vehicle axle loads from bridge dynamic responses. *Journal of Sound and Vibration*, *236*(4), 705–724. doi: 10.1006/jsvi.2000.3021.

[36] Asnachinda, P., Pinkaew, T., Laman, J.A. (Oct. 2008). Multiple vehicle axle load identification from continuous bridge bending moment response. *Engineering Structures*, *30*(10), 2800–2817. doi: 10.1016/j.engstruct.2008.02.018.

[37] Dowling, J., O'Brien, E.J., González, A. (Nov. 2012). Adaptation of Cross Entropy optimisation to a dynamic Bridge WIM calibration problem. *Engineering Structures*, *44*(44), 13–22. doi: 10.1016/j.engstruct.2012.05.047.

[38] Law, S., Bu, J., Zhu, X.Q. (2004). Vehicle axle loads identification using finite element method. *Engineering Structures*, *26*(8), 1143–1153. doi: 10.1016/j.engstruct.2004.03.017.

[39] Viola, P., Jones, M. (Dec. 2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society*

Conference on Computer Vision and Pattern Recognition. CVPR 2001 (p. I–I). doi: 10.1109/CVPR.2001.990517.

[40] Dalal, N., Triggs, B. (Jun. 2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (Vol. 1, pp. 886–893). doi: 10.1109/CVPR.2005.177.

[41] Girshick, R.B., Donahue, J., Darrell, T., Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 580–587).

[42] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y. et al. (Oct. 2016). SSD: Single shot multibox detector. *Presented at the Computer Vision – ECCV 2016, in Lecture Notes in Computer Science* (Vol. 9905). Springer. doi: 10.1007/978-3-319-46448-0_2.

[43] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). *You only look once: Unified, real-time object detection* (p. 788). doi: 10.1109/CVPR.2016.91.

[44] Dai, J., Li, Y., He, K., Sun, J. (2023). R-FCN: Object detection via region-based fully convolutional networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, in NIPS'16* (p. 9). Barcelona, Spain: Curran Associates Inc. https://arxiv.org/abs/1605.06409.

[45] Ekberg, P., Daemi, B., Mattsson, L. (Feb. 2017). 3D precision measurements of meter-sized surfaces using low cost illumination and camera techniques. *Measurement Science and Technology*, *28*, 045403. doi: 10.1088/1361-6501/aa5ae6.

[46] Illingworth, J., Kittler, J. (1987). The adaptive Hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *9*(5), 690–698. doi: 10.1109/TPAMI.1987.4767964.

[47] Winkler, S., Yu, H., Zhou, Z. (Feb. 2007). Tangible mixed reality desktop for digital media management. In A.J. Woods, N.A. Dodgson, J.O. Merritt, M.T. Bolas, I.E. McDowall (Eds.), *Presented at the Electronic Imaging 2007* (p. 64901S). San Jose, CA, USA. doi: 10.1117/12.703906.

[48] Hartley, R.I., Zisserman, A. (2004). *Multiple view geometry in computer vision, second*. Cambridge University Press, ISBN: 0521540518.

[49] Chen, S., Guo, W. (2023). Auto-encoders in deep learning—a review with new perspectives. *Mathematics*, *11*(8), 1777. doi: 10.3390/math11081777.

[50] LeCun, Y., Cortes, C. (2005). *The MNIST database of handwritten digits*. https://api.semanticscholar.org/CorpusID:60282629.

[51] Hebb, D.O. (1949). *The organization of behavior*. Wiley.

[52] Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K. (Aug. 2018). Convolutional neural networks: An overview and application in radiology. *Insights into Imaging*, *9*(4), 611–629. doi: 10.1007/s13244-018-0639-9.

[53] Makone, A. (2020). *The landscape of deep learning based object detection techniques in computer vision*. doi: 10.13140/RG.2.2.12679.42401.

[54] Kingma, D.P., Ba, J. (Jan. 29, 2017). Adam: A method for stochastic optimization. *arXiv: arXiv:1412.6980*. Accessed: Feb. 27, 2024. http://arxiv.org/abs/1412.6980.