Fooling Deep-Learning-Based Bridge Health Monitoring Models: A Hacker Vehicle

Yifu Lan*, Zhenkun Li, Weiwei Lin
Department of Civil Engineering, Aalto University, 02150 Espoo, Finland

ABSTRACT

Despite the widespread adoption of data-driven Structural Health Monitoring (SHM) systems, they may in fact be vulnerable. This paper demonstrates, from an attacker's perspective, how to hack a deep learning (DL)-based bridge SHM system perturbing healthy signals to appear as damaged or vice versa, disguising damaged signals as healthy. They can result in misclassification by the model, thereby fooling the SHM system. Its significance lies not only in raising concerns about the vulnerability of SHM systems but also in giving meaning to a new field, the defence of SHM systems. The paper first proposes a Principal Feature Attack (PFA) algorithm that seeks moderate alterations to the most influential features within physical constraints to attack the model, which is a black-box attack algorithm. It then conceptualizes a hacker vehicle that, by passing over the bridge, indirectly edits the bridge vibration signals through the Vehicle-Bridge Interaction (VBI) process, thus physically and unauthorizedly writing adversarial alterations into the system. Through a typical case study of a traffic event-based bridge SHM, the method's performance is evaluated. The results prove the method to be effective in both nontarget and target attacks on the SHM system, with the modifications being visually similar to the original and comply with physical constraints, making them hard to detect. The general effectiveness of the method is demonstrated through attacks on different DL models, where complex models appear more vulnerable to this threat.

Keywords: structural health monitoring, hacker vehicle, adversarial attack, deep learning, neural network, vehicle-bridge interaction.

1. Introduction

As a crucial component of transportation infrastructure, bridges play a vital role in social and economic development. However, over time, the safety and durability of bridge structures have become a global concern. For instance, in the United States, more than 42% of bridges have been in use for over half a century, with approximately 7.5% exhibiting structural deficiencies [1]. In Europe, most bridges were constructed between 1945 and 1965, many of which are now showing signs of aging and degradation [2]. Therefore, timely assessment of structural conditions to ensure safety is essential. Nonetheless, traditional methods of visual inspections are not only labour-intensive but also incapable of detecting internal damage [3]. To address this issue, SHM approaches based on sensing technologies have been proposed and seen increasing application in bridge structures over the past few decades [4,5].

Vibration-based SHM is an area that has garnered widespread attention. Depending on the damage indicators used, the mainstream methods can be categorized into modal parameter-based methods and data-driven methods. Changes in bridge modal parameters (such as frequency and mode shapes) can indicate structural damage, and these methods are known as modal parameter-based methods [6]. However, many such techniques struggle to detect minor damage, and modal parameter-based methods can be subjective, relying on expert knowledge, and are at risk of human bias, conflicting with the trend of automatic SHM in smart cities [7,8].

In recent years, with the rapid development of artificial intelligence (AI), data-driven methods have sharply become a mainstream area. They typically rely on large datasets and machine learning (ML) models such as Support Vector Machines (SVMs) and Neural Networks (ANNs) [9]. Current research hotspots focus on DL models, with progress chiefly concentrated on improving detection accuracy, developing real-time monitoring algorithms, and training models to distinguish changes related to damage from those caused by temperature fluctuations or other external factors [8]. Although they usually lack detailed physical interpretabilities, their high precision and efficiency have made them

widely sought after [10–15]. They are inevitably integrating into urban infrastructure monitoring, becoming an indispensable part of smart cities.

However, data-driven SHMs, including DL models, can be quite vulnerable. Research in the field of ML has demonstrated that adding imperceptible perturbations to a target objective can render models like ANNs ineffective; this is known as adversarial attacks [16]. In the field of autonomous driving, this could lead to erroneous driving manoeuvres (e.g., due to incorrect traffic sign recognition), resulting in traffic accidents [17]. In the realm of banking asset management, it could lead to misrecognition of client identity, thereby causing financial loss [18]. This is also true in the field of bridge health management; for instance, maliciously classifying a damaged structure as healthy is of serious concern. If undetected, such types of attacks could lead to deterioration in structural health or even critical failures. Similarly, unnecessary maintenance and inspections resulting from falsely labelling a healthy structure as damaged could impose financial toll, and repeated misclassifications could undermine confidence in the monitoring systems [19]. To make matters worse, unlike tasks such as image recognition, many SHM tasks are not intuitive, and it is difficult for humans to identify damage solely by observing the measurement signals from structures. This means that attacks on SHM systems are even harder to detect. Either scenario could potentially cripple the system.

A direct application may be military-purpose SHM attacks, for example, strategically attacking target SHM systems. Notable large-scale system deployments, such as the Integrated Condition Assessment System (ICAS) utilized by the U.S. Navy [20], can be potential targets. However, the authors' intention is primarily to prompt engineers and researchers to be aware of the malicious attack and the vulnerability of SHM system itself, and to adopt appropriate preventative measures before tragedies occur. Recognizing the existence of human attacks, knowing the potential harm they can cause, and understanding of the methodologies of them are the keys to formulating defence strategies. A good defender should think from the perspective of an attacker.

Classic adversarial attacks typically require the attacker to have full access to the system. This may include model structure, parameters, and gradient information, as well as access to the training data, inputs, and outputs; this is known as a white-box attack [21]. Considering the complexity and responsibility of SHM systems, one strategy to guard against sub-standard SHM implementations is to mandate transparency in these systems, whereby their designs should be made publicly available. This makes white-box attacks on SHM systems somewhat feasible, not to mention the insider threats. On the other hand, even if an attacker only has access to the inputs and outputs of an SHM system, without any knowledge of the inner workings of the model, attacks are still possible; these are known as a black-box attack [22]. The methods often involve algorithms seeking sufficiently minor alterations to the model or the samples that can lead to misclassification by the model [16,21]. These are adversarial attacks in the context of data science. If that were the case, exploring these attacks and developing corresponding defence strategies seem to be the tasks of data/computer science researchers.

However, adversarial attacks in SHM engineering are even more complex due to several reasons:

- Physical constraints - In SHM engineering, the uncertainties in data (e.g., environmental noise) may mask minor adversarial alterations; whereas alterations that exceed the physical response to normal events might be immediately discovered (if the system has an alert for it), even though these changes are not visually significant in the data itself. Adversarial modifications need to comply with physical constraints.
- Data modification difficulties - Current adversarial attacks generally assume that the input sample data, like an image to be recognized, can be directly (or conveniently) modified; adversarial alterations can be easily added to the system. However, unlike data/computer science, in SHM engineering, outside attackers do not have the authority to directly insert/alter sample data, as the model acquires them straight from the structural responses.

This paper demonstrates, from an attacker's perspective, how to hack a data-driven bridge SHM system (a DL network in this study). It attempts to fool an event-based bridge monitoring system, where sensors on the bridge collect vibration data from the bridge structure, induced by traffic events.

The ML model is trained through a dataset established from the collected data and predicts newly acquired data. This is a common practice in bridge SHM, such as the monitoring of the Sydney Bridge by Diez et al [23]. The authors proposed an attack algorithm targeting principal features (PFs) and a hacker vehicle as a tool for "editing" bridge signals, perturbing healthy signals to appear as damaged or vice versa, disguising damaged signals as healthy. They can result in misclassification by the model, thereby fooling the SHM system. The contributions of this paper are mainly in the following areas:

- Proposal of a novel black-box PFA algorithm - It first eavesdrops on the model, identifying the features that most significantly impact the results (i.e., PFs) through feature perturbation, and then employs the Particle Swarm Optimization (PSO) algorithm to seek PF alterations that maximize the model's prediction error within physical constraints.
- Conceptualization of a hacker vehicle - Based on the results of the PFA algorithm, an exciter is designed and installed on a vehicle. As it passes over the bridge, it influences bridge vibrations through the VBI process, thereby indirectly editing the bridge vibration signals. It unauthorizedly writes adversarial alterations into the system.
- Demonstration of the method's effectiveness - Through a case of data-driven bridge SHM, it demonstrates how to physically hack an SHM system using the proposed method. To the best of the authors' knowledge, this is the first attempt at either computer science or civil engineering. Its significance lies not only in raising concerns about the vulnerability of SHM systems but also in giving meaning to a new field, the defence of SHM systems.

2. A case of data-driven bridge SHM

2.1. Event-based bridge SHM

This section introduces a typical data-driven event-based bridge SHM system as illustrated in **Fig. 1**. It comprises three key components: data collection, model training, and assessment of structural condition. The structural responses caused by passing vehicles (referred to events) are directly collected from the mid-span sensor on a simply supported bridge. In the data collection phase, a data truncation window is employed to capture a 5-second data segment following a vehicle's entry onto the bridge. Data segments shorter than 5 seconds are automatically zero-padded to maintain consistent length. To guarantee the model's robustness against diverse vehicles and noise, an ample and varied dataset is amassed. This dataset encompasses traffic events that cover a wide range of vehicle parameters and environmental noises. The data are then pre-processed to transform into frequency domain data, forming a group of vectors that can be fed into an ML model. These vectors are stored in dataset $\boldsymbol{\mathcal{D}} = [\boldsymbol{d_1}, \boldsymbol{d_2}, \boldsymbol{d_3}, \ldots, \boldsymbol{d_{nt}}]$, where nt is the number of traffic events. In this study, attention will be given to frequency domain signals, as humans can more intuitively discern changes in structural conditions from frequency, compared to time domain signals.

In the model training phase, this study opts for the popular Convolution Neural Network (CNN) models, particularly the 1D-CNN model, which is considered one of the most effective neural networks for extracting damage-sensitive features from signals [24]. A typical CNN comprises convolutional layers, pooling layers, a flatten layer, a connected layer, and a softmax output layer [25]. The softmax layer is utilized to form different probability values of being different classes, as described in Equation (1). The detailed configuration of the model and the training process will be discussed in Section 2.3.

$$P_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \#(1)$$

where z_i represents the input to the softmax layer, and P_i denotes the probability of the sample belonging to the i-th class with a total of C classes.

In the condition assessment phase, data $\boldsymbol{x}$ from a new traffic event, after preprocessing, is input into the trained model $\mathcal{M}$. It produces a probability vector $\boldsymbol{y}$. This output vector $\boldsymbol{y}$ is of the form:

$$\boldsymbol{y} = \mathcal{M}(\boldsymbol{x}) = \{P_1, P_2, P_3, \ldots, P_C\} \#(2)$$

where each element P_i in the vector $\boldsymbol{y}$ represents the probability that the input $\boldsymbol{x}$ belongs to the i-th label out of a total of C labels. The model's prediction is determined by the label corresponding to the highest probability in the $\boldsymbol{y}$.

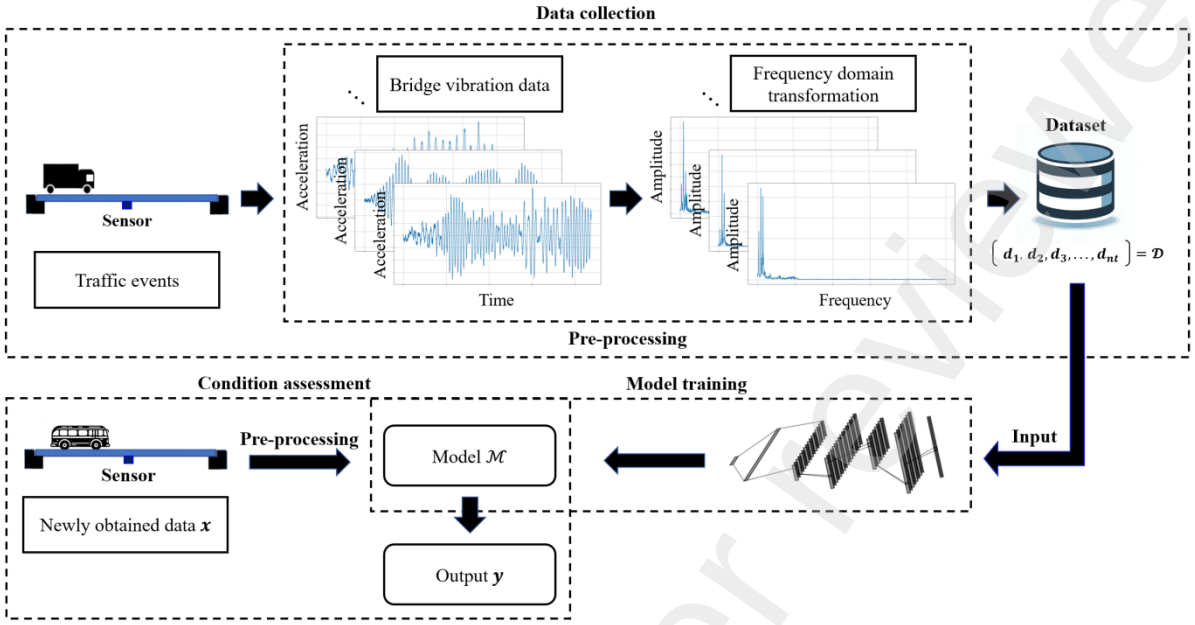


Fig. 1 Data-driven SHM framework.

2.2. VBI model and dataset

2.2.1 VBI model

The SHM system will be demonstrated on a simulated dataset. The following briefly introduces the simulation of bridges and vehicles, as well as the bridge vibrations induced by traffic events (the passages of vehicles). For the bridge model, it is simulated using a simply supported Euler-Bernoulli (EB) beam. Its finite element (FE) model consists of nodes, each with two degrees of freedom (DOF): vertical translation and rotation. The model consists of n elements, $n+1$ nodes, and $2n$ DOFs (excluding the vertical constraints at both ends). The length of the bridge is L, with a uniform flexural rigidity of EI and a mass per unit length of m. Additionally, the damping of the bridge is approximated by mass-stiffness proportional Rayleigh damping. As for the vehicle model, it is simulated by 2-DOF quarter-car model (or representing the vehicle's rear axle system), a simplification has been adopted in many previous studies [26–29]. As illustrated in **Fig. 2**, the VBI process for a traffic event, a car traversing the bridge, is governed by the following equations:

$$\boldsymbol{M_v}\ddot{\boldsymbol{s}}_{\boldsymbol{v}} + \boldsymbol{C_v}\dot{\boldsymbol{s}}_{\boldsymbol{v}} + \boldsymbol{K_v}\boldsymbol{s_v} = \boldsymbol{f_{cv}} \#(3)$$

$$\boldsymbol{M_b}\ddot{\boldsymbol{s}}_{\boldsymbol{b}} + \boldsymbol{C_b}\dot{\boldsymbol{s}}_{\boldsymbol{b}} + \boldsymbol{K_b}\boldsymbol{s_b} = \boldsymbol{f_{cb}} \#(4)$$

Equations (3) and (4) represent the equations of motion pertinent to the vehicle and the bridge, respectively. The matrices $\boldsymbol{M_v}$, $\boldsymbol{C_v}$, and $\boldsymbol{K_v}$ are indicative of the mass, damping, and stiffness of the vehicle, while $\boldsymbol{M_b}$, $\boldsymbol{C_b}$, and $\boldsymbol{K_b}$ denote the mass, damping, and stiffness matrices for the bridge. The terms $\boldsymbol{s_v}$ and $\boldsymbol{s_b}$ signify the displacement vector of the vehicle and the nodal displacement of the bridge system, respectively. Moreover, $\boldsymbol{f_{cv}}$ and $\boldsymbol{f_{cb}}$ are the time-dependent interaction forces exerted on the vehicle and the bridge, respectively.

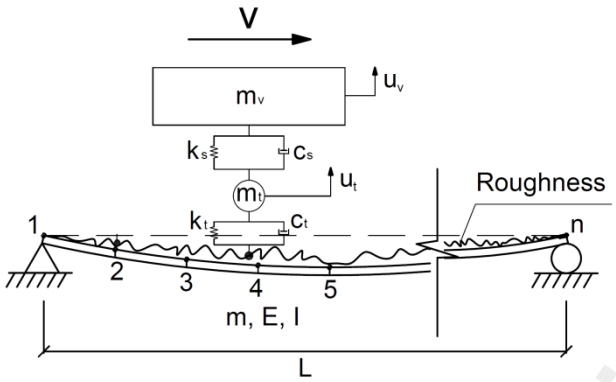


Fig. 2 VBI model.

For the vehicle model in this study, the body and axle masses are represented by m_v and m_t, the suspension and tire damping are indicated by c_s and c_t, and the suspension and tire stiffnesses are denoted by k_s and k_t. The vertical displacements of the vehicle body and axle are symbolized by u_v and u_t, respectively.

$$\boldsymbol{M_v} = \begin{bmatrix} m_v & \\ & m_t \end{bmatrix} \#(5)$$

$$\boldsymbol{C_v} = \begin{bmatrix} c_s & -c_s \\ -c_s & c_s + c_t \end{bmatrix} \#(6)$$

$$\boldsymbol{K_v} = \begin{bmatrix} k_s & -k_s \\ -k_s & k_s + k_t \end{bmatrix} \#(7)$$

$$\boldsymbol{s_v} = \{u_v \quad u_t\}^T \#(8)$$

Road roughness is generated in accordance with ISO 8608 [30], with the roughness coefficient $G_d(n_{s,0}) = 16 \times 10^{-6}\, m^3$ (Class A). It is noteworthy that in the simulations of this study, the road roughness for each traffic event varies (unfixed random seed), reflecting the realistic scenario where different vehicles traverse varying trajectories and road surfaces. Furthermore, 5% Gaussian noise is added to the vibration data to simulate environmental effects [31]. The VBI process is solved using the Newmark-Beta method ($\beta = 0.25$, $\gamma = 0.5$). Further details about road roughness, noise, and the VBI process can be referred to the reference [27,32].

2.2.2 Dataset establishment

In this study, the bridge's parameters are as follows: mass per unit length $m = 2400\, kg/m$, flexural rigidity $EI = 5.5 \times 10^9\, N \cdot m^2$, length $L = 25\, m$. It is divided into 10 elements ($n = 10$), with each length $l = 2.5\, m$ (see **Fig. 3**). These parameters only present a short-span bridge example here, but the methodological framework is not limited to a specific bridge. Generally, bridge damage can be considered as a loss of stiffness [33], as shown in Equation (9). This can be used to represent the bridge damage such as cracks and delamination [34]. In the equation, R_D^i represents the damaged element; R_H^i denotes the intact element; μ represents the reduction coefficient in the i-th element (i.e., damage location).

$$R_D^i = \mu \times R_H^i \#(9)$$

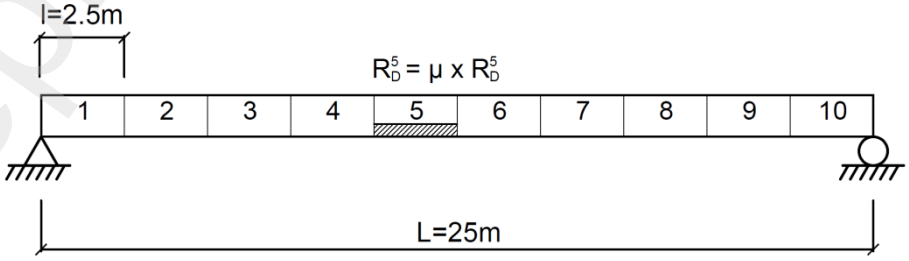


Fig. 3 Bridge model.

Traffic events should be diverse. There is a vehicle passing through the bridge (e.g., a minibus) with the following parameters: $m_v^p = 1.28 \times 10^4\, kg$, $m_t^p = 1.0 \times 10^3\, kg$, $c_s^p = 1.0 \times 10^3\, N \cdot s/m$, $c_t^p = 0$, $k_s^p = 4.0 \times 10^5\, N/m$, $k_t^p = 3.5 \times 10^5\, N/m$, and $v^p = 8\, m/s$. Assuming the parameters of the cars passing over the bridge follow a normal distribution corresponding to this vehicle, its parameters for the i-th traffic event would be:

$$\boldsymbol{V^i} \sim \mathcal{N}\left\{\boldsymbol{V^p}, \mathrm{diag}\left[\left(\frac{\boldsymbol{V^p}}{4}\right)^2\right]\right\} \#(10)$$

where $\boldsymbol{V^i} = \left\{m_v^i, m_t^i, c_s^i, c_t^i, k_s^i, k_t^i, v^i\right\}^T$, and $\boldsymbol{V^p} = \left\{m_v^p, m_t^p, c_s^p, c_t^p, k_s^p, k_t^p, 0.4v^p\right\}^T$. $\mathcal{N}$ denotes a multivariate normal distribution. The relatively small variance in speed is attributed to the fact that bridge traffic typically adheres to speed limits. Additionally, each traffic event incorporates 5% environmental noise and random road roughness to ensure the robustness of the results against these factors. The sampling rate is 1000 Hz (or the time step is 0.001 s). For different damage cases (DCs), each DC consists of 1000 traffic events, forming a database. Details of this can be referenced in **Table 1**. They are then randomly divided into training and testing sets at a 9:1 ratio.

Table 1 Dataset of DCs

DCs	Label	Description	Runs
DC 0	0	Healthy	1000
DC 1	1	Damage on 5-th element, μ = 0.6	1000
DC 2	2	Damage on 5-th element, μ = 0.7	1000
DC 3	3	Damage on 5-th element, μ = 0.8	1000
DC 4	4	Damage on 5-th element, μ = 0.9	1000
DC 5	5	Damage on 1-st element, μ = 0.5	1000
DC 6	6	Damage on 3-rd element, μ = 0.5	1000
DC 7	7	Damage on 5-th element, μ = 0.5	1000
DC 8	8	Damage on 7-th element, μ = 0.5	1000
DC 9	9	Damage on 9-th element, μ = 0.5	1000

2.3. DL model and performance

As labels are available for the dataset, the CNN model can be trained in a supervised manner. After some trials by the authors, a 1D-CNN architecture that performs well for this study's dataset is illustrated in **Table 2**. Notably, it utilizes LeakyReLU instead of ReLU as the activation function for convolutional layers, which has been shown to avoid neuron death and gradient vanishing issues, thereby enhancing the model's accuracy [35]. Its function is defined as follows (negative slope=0.001):

$$\text{LeakyReLU}\,(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0.001x, & \text{otherwise} \end{cases} \#(11)$$

Table 2 CNN configurations

Layer	Output shape	Parameter	Activation
Conv1d	2500 × 64	Kernel number: 64; Kernel size:10; Stride: 1; Padding: “same”	LeakyReLU
Max pooling	1250 × 64	Kernel: 2; Stride: 2	None
Conv1d	1250 × 128	Kernel number: 128; Kernel size:10; Stride: 1; Padding: “same”	LeakyReLU
Max pooling	625 × 128	Kernel: 2; Stride: 2	None
Flatten	80000	None	None
Dense	30	None	LeakyReLU
Dense	10	None	Softmax

The model training was conducted in a Python 3.11 environment using TensorFlow [36]. In addition to the above architecture, the hyperparameters guiding the behaviour of the CNN were chosen in this manner: a batch size was set at 32, Adam was used as the optimizer, the learning rate was 1×10^{-4}, the employed loss function was "Cross-Entropy Loss", and the total number of training epochs was 200. The accuracy can be calculated by [37]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \#(12)$$

where TP, FP, TN, and FN represent "True Positive", "False Positive", "True Negative", and "False Negative", respectively. The loss and accuracy during the model's training and testing phases are illustrated in **Fig. 4** (i.e., training history). Evidently, the CNN achieved relatively high testing accuracy (> 0.9). It stabilized around 50 epochs, and based on the test loss trend, there are no apparent overfitting issues.

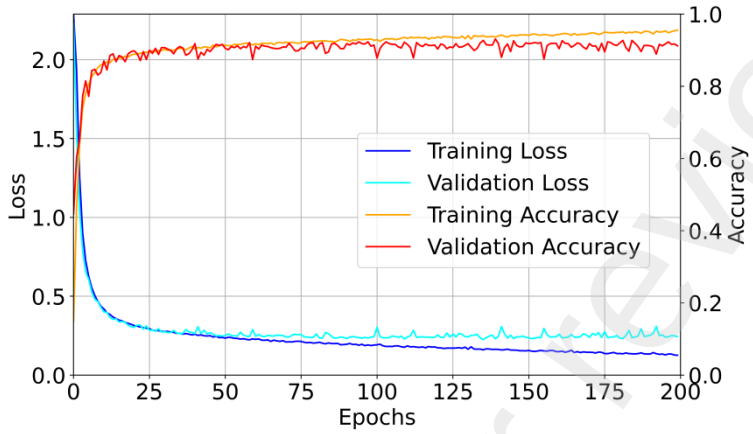


Fig. 4 Training history for CNN.

Fig. 5 indicates that the model's suboptimal predictions are primarily on distinguishing between small-scale damage (DC 4) and its adjacent state (DC 3), as well as the healthy state (DC 0), due to their high similarity in signals. Nonetheless, the accuracy in these cases remains above 75%. For the other DCs, model $\mathcal{M}$ performs quite well (e.g., 100% accuracy on DC 6 and DC 9). The overall accuracy stands at 91.2%.

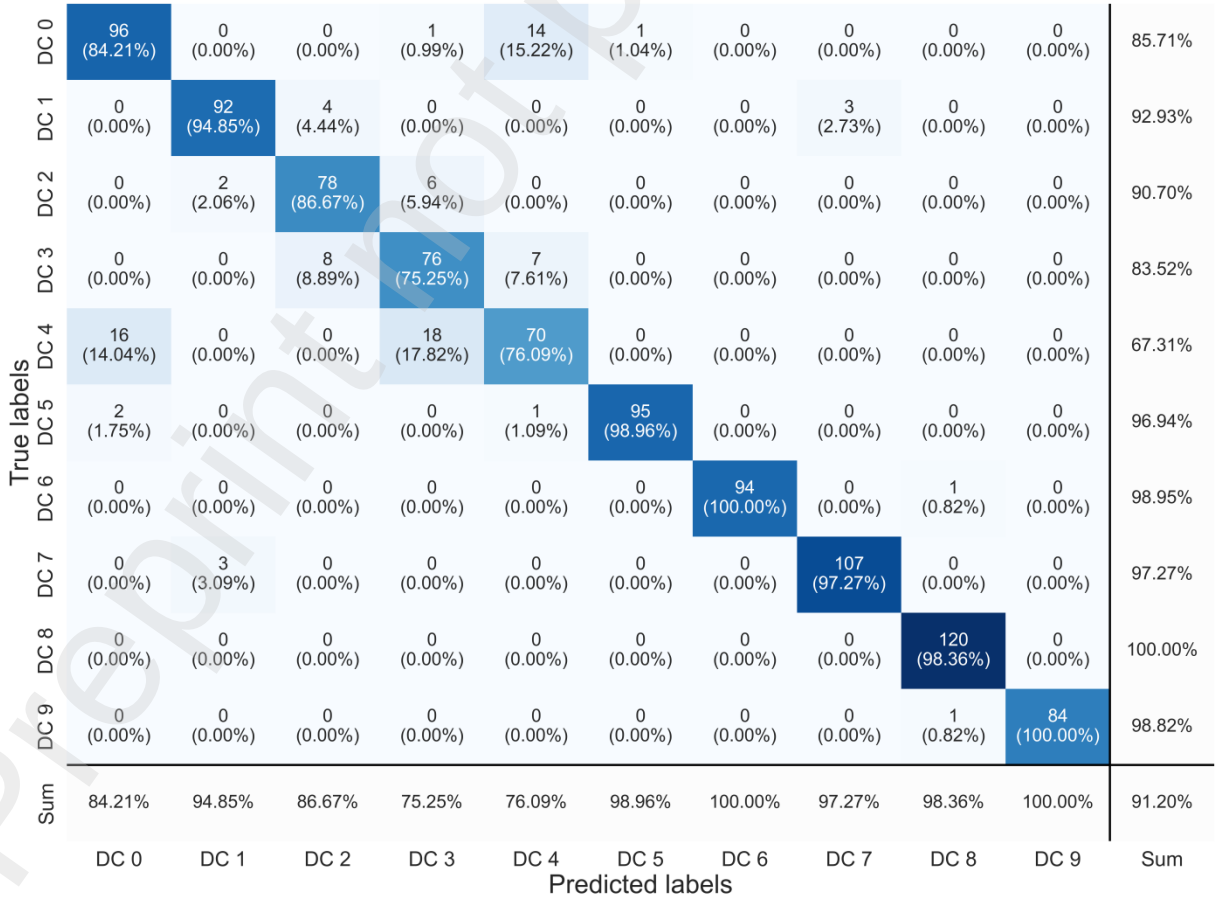


Fig. 5 Confusion matrix.

3. Attack on SHM

In this section, we will present the proposed PFA algorithm and the conceptualization of the hacker vehicle, aiming at fooling an SHM system. The innovation of PFA algorithm lies in its approach of seeking moderate alterations to the first few features that have the greatest impact on the model results, rather than looking for the smallest modification in every feature as many algorithms in computer science do [16]. Because minimal modifications can be masked by physical uncertainties such as noise etc. It also diverges from methods like the One-pixel attack [38], which makes substantial changes to a single feature, changes that can easily be detected as anomalous data; their modifications should comply with physical constraints (in this case, less than the physical response from normal events). Moreover, the method is simple and efficient. The concept of the hacker vehicle here is that it acts as an editing tool for bridge vibration data, indirectly writing targeted alterations into the samples; this gives a new idea for unauthorized outsider attack.

3.1. PFA algorithm

The attack on the SHM system is conducted in a black-box manner. This means, for model $\mathcal{M}$ trained above, information such as training history or gradients is not available, and the model can only be accessed on an input–output basis. This is a scenario of an outsider hacking an SHM system. The first step is to eavesdrop on the model, that is, to listen each signal input and its output $\boldsymbol{y} = \mathcal{M}(\boldsymbol{x})$. It should be noted that here $\boldsymbol{x}$ is the pre-processed frequency domain data. Let $\boldsymbol{x_i}[f_j]$ denote the frequency f_j of the i-th signal vector. Subsequently, the envelope of the observed traffic events can be listened, determined by two vectors $\boldsymbol{env_{max}}$ and $\boldsymbol{env_{min}}$, each with a frequency range of 0-500 Hz ($j = 1, 2, 3, ..., 2500$), where:

$$\boldsymbol{env_{max}}[f_j] = max\left(\{\boldsymbol{x_i}[f_j]\}_{i=1}^{q}\right) \#(13)$$

$$\boldsymbol{env_{min}}[f_j] = min\left(\{\boldsymbol{x_i}[f_j]\}_{i=1}^{q}\right) \#(14)$$

The second step is to identify the PFs, through feature perturbation. The proposed PFs pick-up algorithm is detailed in **Algorithm 1**: for each frequency (feature), f_j, of the i-th sample, $\boldsymbol{x_i}$, it adds Gaussian noise with a standard deviation of η into the amplitude, $\boldsymbol{x_i}'[f_j]$, and captures the model's prediction, $\boldsymbol{y_i}'$. The impact of each feature is quantified by the absolute change in model prediction, $\boldsymbol{I_i}$, and averaged over the perturbed samples, $\boldsymbol{I^{avg}}$. Features are then ranked based on their impact, $\boldsymbol{I^{idx}}$, and the top T features are encored into an array, $\boldsymbol{p} = \{f_1^{PF}, f_2^{PF}, f_3^{PF}, ..., f_T^{PF}\}$.

Algorithm 1 PFs pick-up algorithm

Require: Model $\mathcal{M}$ and samples $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, ..., \boldsymbol{x_q}]$
Require: Original prediction $\boldsymbol{Y} = \mathcal{M}(\boldsymbol{X}) = [\boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{y_3}, ..., \boldsymbol{y_q}]$
Require: Noise level η (0.001)
$\boldsymbol{I} \leftarrow []$
For sample $\boldsymbol{x_i}$ in $\boldsymbol{X}$:
    $\boldsymbol{I_i} \leftarrow \{\}$
    For f_j in $\boldsymbol{x_i}$:
        $\boldsymbol{x_i}'[f_j] \leftarrow \boldsymbol{x_i}[f_j] + \mathcal{N}(0, \eta^2)$
        $\boldsymbol{y_i}' \leftarrow \mathcal{M}(\boldsymbol{x_i}')$
        $\boldsymbol{I_i} \leftarrow \boldsymbol{I_i} \cup \left(|\boldsymbol{y_i} - \boldsymbol{y_i}'|\right)$
    $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \boldsymbol{I_i}$
$\boldsymbol{I^{avg}} \leftarrow mean(\boldsymbol{I})$
$\boldsymbol{I^{idx}} \leftarrow argsort(-\boldsymbol{I^{avg}})$
$\boldsymbol{p} \leftarrow \boldsymbol{I^{idx}}[:T]$
Return $\boldsymbol{p}$

The third step is to seek PF alterations that maximize the model's prediction errors using the PSO optimizer, where physical constraints are imposed. PSO is an optimization technique inspired by the social behaviours of birds and fish, particularly used for finding optimal solutions within a defined search space. Detailed information about the PSO method can be found in reference [39]. Attacks on PFs can be categorized into nontarget and target attacks. Nontarget attacks aim to cause the model to make any incorrect classification, without specifying what the incorrect classification should be. The goal is simply to ensure the model does not produce the correct result. Target attacks aim to mislead the model into classifying an input as a specific, incorrect category. A typical nontarget attack can fool an SHM system to diagnose a healthy bridge as damaged. The proposed nontarget attack algorithm is described in **Algorithm 2**: For a sample from vehicle#0, denoted as $\boldsymbol{x_0}$, with the configuration of $\boldsymbol{V^0}$ ($\boldsymbol{V^0}\sim\mathcal{N}\left\{\boldsymbol{V^p},\mathrm{diag}\left[\left(\frac{\boldsymbol{V^p}}{4}\right)^2\right]\right\}$), construct the objective function $F(\boldsymbol{x})=\mathcal{M}(\boldsymbol{x_0}')_{Lab\_ori}$, where $\mathcal{M}$ is the model, and for $f_i^{PF}\in\boldsymbol{p}$, $\boldsymbol{x_0}'[f_i^{PF}]=\boldsymbol{x}$, while all other components remain unchanged. Lab_ori represents its original label. Physical constraints in this study are the envelope of observed traffic events $\boldsymbol{env_{max}}$, $\boldsymbol{env_{min}}$, ensuring that the alterations are within the normal physical responses. It should be noted that considering an exciter will be used to physically reproduce these alterations below, it cannot achieve negative vibration changes. Thus, $max(\boldsymbol{env_{min}}[f_i^{PF}],\boldsymbol{x_0}[f_i^{PF}])$ should be adopted as the physical lower limit, $\boldsymbol{lb}$. PSO seeks the optimal solution $\boldsymbol{x_{opt}}=\{\boldsymbol{x_0}'[f_1^{PF}],\boldsymbol{x_0}'[f_2^{PF}],\boldsymbol{x_0}'[f_i^{PF}],...,\boldsymbol{x_0}'[f_T^{PF}]\}$ within these constraints to minimize the prediction probability of the original label:

$$\min_{\{f_i^{PF}\in\boldsymbol{p}\}} P_{Lab\_ori}\left(\mathrm{M}(\boldsymbol{x_0})=\boldsymbol{y_0}|\boldsymbol{x_0}'[f_i^{PF}]\right) \#(15)$$

Algorithm 2 Nontarget attack on PFs

Require: Model $\mathcal{M}$
Require: Sample from the vehicle#0 $\boldsymbol{x_0}$
Require: Envelope of observed traffic events $\boldsymbol{env_{max}}$, $\boldsymbol{env_{min}}$
Require: PSO optimizer
Require: PFs $\boldsymbol{p}$
Define the objective function $F(\boldsymbol{x})=\mathcal{M}(\boldsymbol{x_0}')_{Lab\_ori}$:
    $\boldsymbol{x_0}'[f_i^{PF}]=\boldsymbol{x}\ for\ f_i^{PF}\in\boldsymbol{p}$
    $\boldsymbol{x_0}'[f_i]=\boldsymbol{x_0}[f_i]\ for\ f_i\notin\boldsymbol{p}$
Physical constraints
$\boldsymbol{lb}\leftarrow max(\boldsymbol{env_{min}}[f_i^{PF}],\boldsymbol{x_0}[f_i^{PF}])\ for\ f_i^p\in\boldsymbol{p}$
$\boldsymbol{ub}\leftarrow\boldsymbol{env_{max}}[f_i^{PF}]\ for\ f_i^{PF}\in\boldsymbol{p}$
Execute PSO to find $\boldsymbol{x_{opt}}$:
    $\boldsymbol{x_{opt}}\leftarrow PSO[F(\boldsymbol{x}),\boldsymbol{lb},\boldsymbol{ub}]$
Return $\boldsymbol{x_{opt}}$

For a target attack, a typical case is to fool an SHM system to diagnose a damaged bridge as healthy, which could pose a greater risk than the above. The proposed target attack algorithm is described in **Algorithm 3**: The difference between it and **Algorithm 2** is that its objective function is $-F(\boldsymbol{x})=\mathcal{M}(\boldsymbol{x_0}')_{Lab\_tar}$, where Lab_tar is the target fake label. The purpose is to seek the optimal solution $\boldsymbol{x_{opt}}$ to maximize the prediction probability of model $\mathcal{M}$ for Lab_tar:

$$\max_{\{f_i^{PF}\in\boldsymbol{p}\}} P_{Lab\_tar}\left(\mathrm{M}(\boldsymbol{x_0})=\boldsymbol{y_0}|\boldsymbol{x_0}'[f_i^{PF}]\right) \#(16)$$

Algorithm 3 Target attack on PFs

Require: Model $\mathcal{M}$
Require: Sample from the vehicle#0 $\boldsymbol{x_0}$
Require: Envelope of observed traffic events $\boldsymbol{env_{max}}$, $\boldsymbol{env_{min}}$
Require: PSO optimizer
Require: PFs $\boldsymbol{p}$
Require: Target label index Lab_tar
Define the objective function $F(\boldsymbol{x}) = -\mathcal{M}(\boldsymbol{x_0}')_{Lab\_tar}$:
$\boldsymbol{x_0}'[f_i^{PF}] = \boldsymbol{x}\ for\ f_i^{PF} \in \boldsymbol{p}$
$\boldsymbol{x_0}'[f_i] = \boldsymbol{x_0}[f_i]\ for\ f_i \notin \boldsymbol{p}$
Physical constraints
$\boldsymbol{lb} \leftarrow max(\boldsymbol{env_{min}}[f_i^{PF}], \boldsymbol{x_0}[f_i^{PF}])\ for\ f_i^p \in \boldsymbol{p}$
$\boldsymbol{ub} \leftarrow \boldsymbol{env_{max}}[f_i^{PF}]\ for\ f_i^{PF} \in \boldsymbol{p}$
Execute PSO to find $\boldsymbol{x_{opt}}$:
$\boldsymbol{x_{opt}} \leftarrow PSO[F(\boldsymbol{x}), \boldsymbol{lb}, \boldsymbol{ub}]$
Return $\boldsymbol{x_{opt}}$

3.2. Hacker vehicle

The concept of a hacker vehicle is here introduced, which, based on the results of the PFA algorithm, has an exciter designed and installed on it (see **Fig. 6**). As the vehicle passes over the bridge, it influences bridge vibrations through VBI process, thereby physically and unauthorizedly writing adversarial alterations into the system. Another advantage of using a hacker vehicle is that it is sufficiently concealed. For example, directly exciting the bridge with an on-site vibrator would be an action easily discovered.

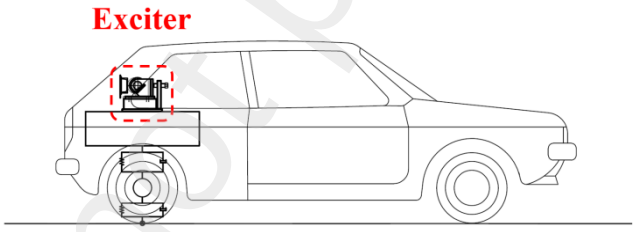


Fig. 6 Exciter on the vehicle.

For the optimal solution $\boldsymbol{x_{opt}}$ of PFs, its original values $\boldsymbol{x_0}[\boldsymbol{p}]$, and the upper physical limit $\boldsymbol{env_{max}}[\boldsymbol{p}]$, we can define an index vector $\boldsymbol{\chi}$, as shown in Equation (17), which has a length of $|\boldsymbol{p}|$. We then select the elements whose values are 1 from $\boldsymbol{\chi}$ to form a new vector $\boldsymbol{\kappa}$ (see Equation (18)). This is because minor adversarial changes would be obscured by the uncertainties of the physical process itself, and we select the largest alterations permissible within physical constraints. $\boldsymbol{\kappa}$ is the target frequencies of action for the exciter, and $\boldsymbol{x_{opt}}[\boldsymbol{\kappa}]$ represents the target amplitudes to be reconstructed using the exciter.

$$\boldsymbol{\chi} = \frac{\boldsymbol{x_{opt}} - \boldsymbol{x_0}[\boldsymbol{p}]}{\boldsymbol{env_{max}}[\boldsymbol{p}] - \boldsymbol{x_0}[\boldsymbol{p}]} \#(17)$$

$$\boldsymbol{\kappa} = \{f_q | \boldsymbol{\chi}[f_q] = 1, f_q \in \boldsymbol{p}\} \#(18)$$

The vibration of the exciter is controlled by Equation (19). In the equation, $\boldsymbol{A}$ represents the set amplitudes corresponding to $\boldsymbol{\kappa}$.

$$\boldsymbol{\Gamma} = \boldsymbol{A}\ cos(2\pi\boldsymbol{\kappa}t)\ \#(19)$$

$\boldsymbol{\Gamma}$ can be incorporated into the VBI process as part of the vehicle force. According to the authors' trials, the amplitudes at the target frequencies of the bridge $\boldsymbol{x_\kappa}$ exhibits a rough linear relationship with the set amplitudes $\boldsymbol{A}$, which will be shown in the following sections. It can be described as:

$$\boldsymbol{x_\kappa} \approx \boldsymbol{\beta_0} + \boldsymbol{\beta_1 A} + \boldsymbol{\varepsilon} \#(20)$$

where $\boldsymbol{\beta_0}$ is the y-intercept vector of the regression, $\boldsymbol{\beta_1}$ is the slope vector of the regression, and $\boldsymbol{\varepsilon}$ is the random error term, capturing deviations from the linear model due to uncertainties like road roughness and noise. They can be estimated using least squares regression.
So, $\boldsymbol{A}$ can be calculated as:

$$\boldsymbol{A} \approx \frac{\boldsymbol{x_{opt}[\kappa]} - \boldsymbol{\beta_0} - \boldsymbol{\varepsilon}}{\boldsymbol{\beta_1}} \#(21)$$

So far, a hacker vehicle equipped with an exciter, governed by the vibration function $\boldsymbol{\Gamma}$, is set up. The following section will demonstrate its performance in fooling the SHM system. The process of attacking the SHM system is summarised in **Fig. 7**.

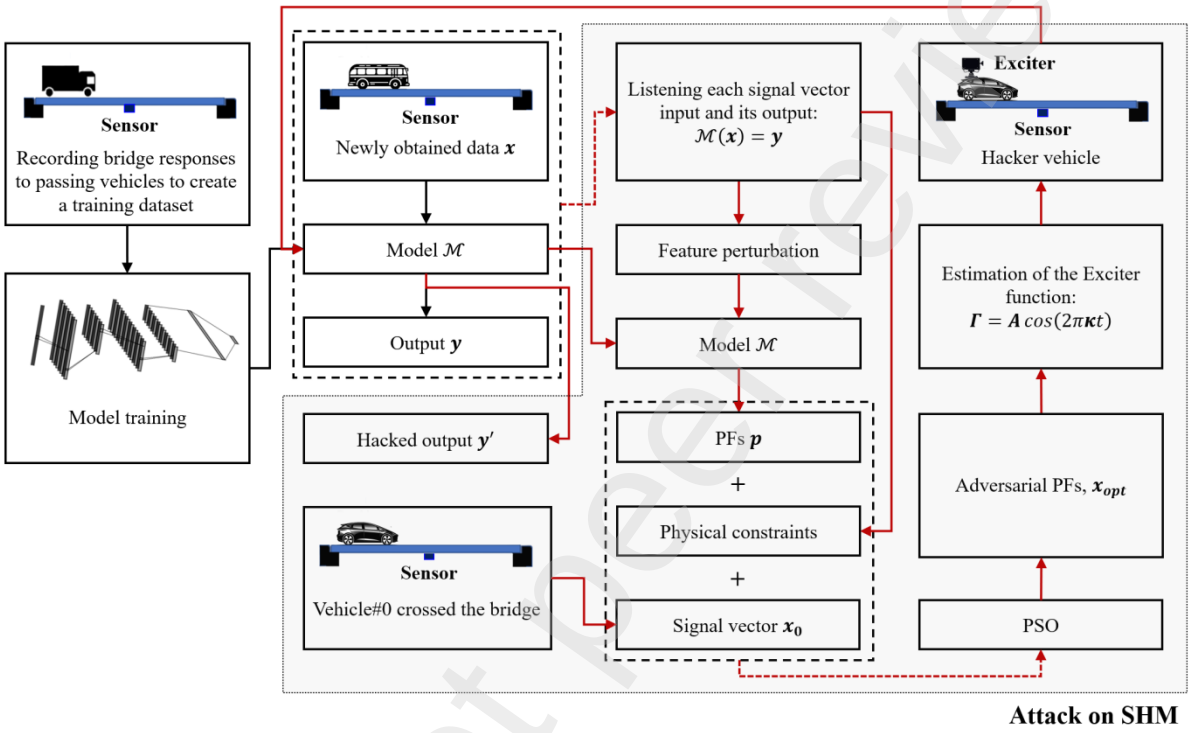


Fig. 7 Attack on SHM.

4. Evaluation and results

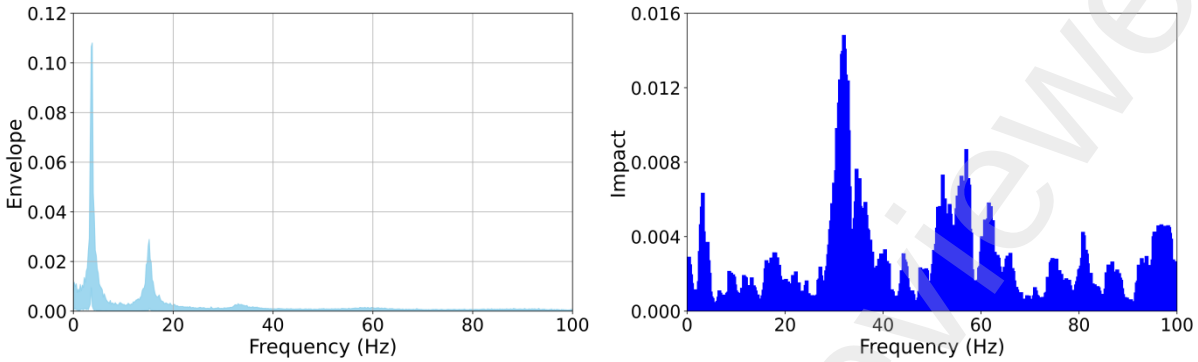
The evaluation of the proposed attack method is based on the dataset acquired on Section 2. Attempts will be made to conduct both nontarget and target attacks, demonstrating how they hack a SHM system, as well as to explore the method's feasibility to different models.

4.1. Nontarget attack

4.1.1 Demonstration of PFA algorithm

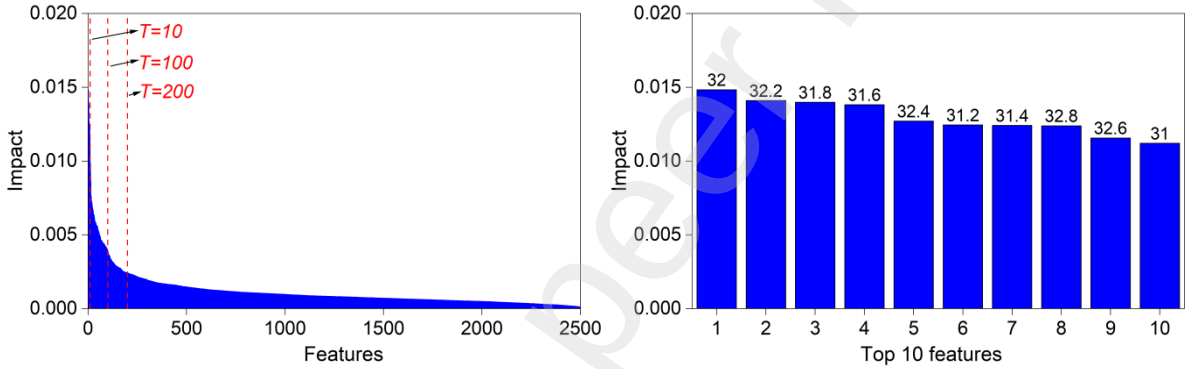
Nontarget attacks will be first on health status (DC 0), fooling the SHM system to diagnose a healthy sample as damaged. By listening to 100 samples from normal DC 0, an envelope graph, as illustrated in **Fig. 8a** (showing content from 0-100 Hz), can be obtained; it represents the boundary of bridge responses to normal traffic events. Meanwhile, based on these samples, the impact of frequency (feature) can be obtained using **Algorithm 1**, as shown in **Fig. 8b**. Note that the figure only displays content from 0-100 Hz, as other frequency features have minor impact. From a physical perspective, the first four frequencies of the bridge are respectively 3.8, 15.2, 34.2, and 60.8 Hz. The figure reveals that features near these frequencies indeed receive more attention, but beyond these, many features are also given significant weights. This suggests that, on the one hand, ML models may be able to detect features other than physical modes, thereby having higher sensitivity to damage. On the other hand, the significance of these features may be worth exploring, as it relates to the model's trustworthiness.

As shown in **Fig. 9a**, among all 2500 features corresponding to frequencies, only a few have a significant impact on the model result; they are the PFs we are looking for. As illustrated in **Fig. 9b**, by selecting the top ten influential features ($T = 10$), PFs can be obtained $\boldsymbol{p} = \{32\ Hz, 32.2\ Hz, 31.8\ Hz, 31.6\ Hz, 32.4\ Hz, 31.2\ Hz, 31.4\ Hz, 32.8\ Hz, 32.6\ Hz, 31\ Hz\}$.



(a) (b)

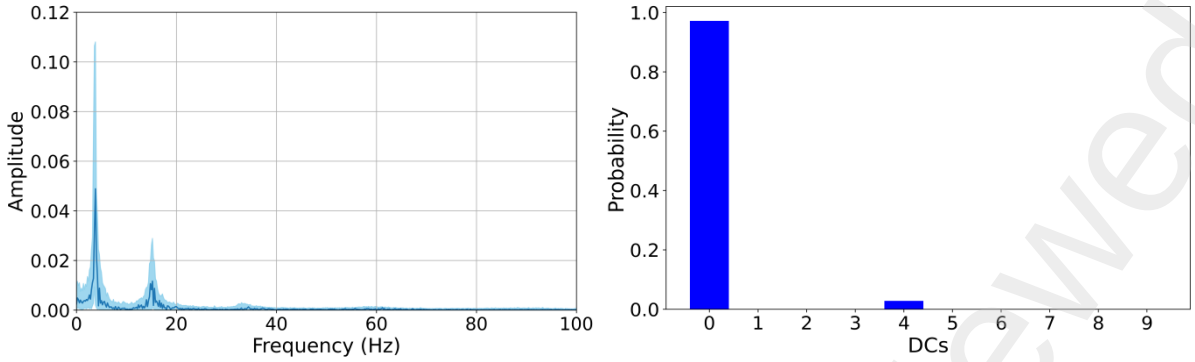
Fig. 8 Attack on DC 0: (a) envelope of observed traffic events, (b) feature impacts.



(a) (b)

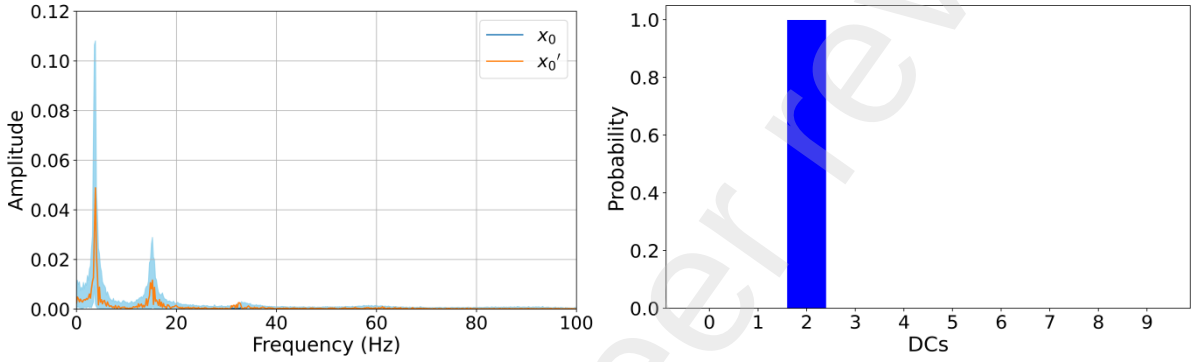
Fig. 9 Selections of PFs: (a) all features, (b) top 10 features.

For the employed vehicle#0 ($\boldsymbol{V^0} \sim \mathcal{N}\left\{\boldsymbol{V^p}, \text{diag}\left[\left(\frac{\boldsymbol{V^p}}{4}\right)^2\right]\right\}$), its configuration is $m_v^0 = 1.28 \times 10^4\ kg$, $m_t^0 = 1.0 \times 10^3\ kg$, $c_s^0 = 1.0 \times 10^4\ N \cdot s/m$, $c_t^0 = 0$, $k_s^0 = 4.0 \times 10^5\ N/m$, $k_t^0 = 3 \times 10^5\ N/m$, and $v^0 = 8\ m/s$. Utilizing the signal of $\boldsymbol{V^0}$ as input, the frequency response of the bridge and the model prediction can be obtained, as shown in **Fig. 10a** and **Fig. 10b**. It is observed that based on normal events as $\boldsymbol{V^0}$ passes over the bridge, the model $\mathcal{M}$ predicts with a 97.2% probability that the bridge condition is DC 0, i.e., the healthy state. However, for the adversarial result of 10 PFs using **Algorithm 2**, despite its frequency response being highly similar to the original, the model predicts with nearly 100% probability that it is DC 2 (damage on 5-th element, μ=0.7), as shown in **Fig. 11a** and **Fig. 11b**. This indicates the nontarget attack has worked very well.



(a) (b)

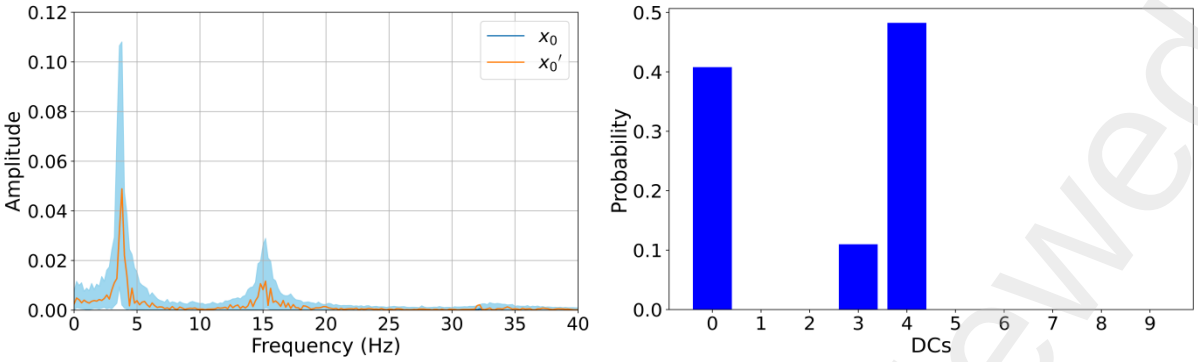
Fig. 10 Original results: (a) frequency response, (b) model prediction.



(a) (b)

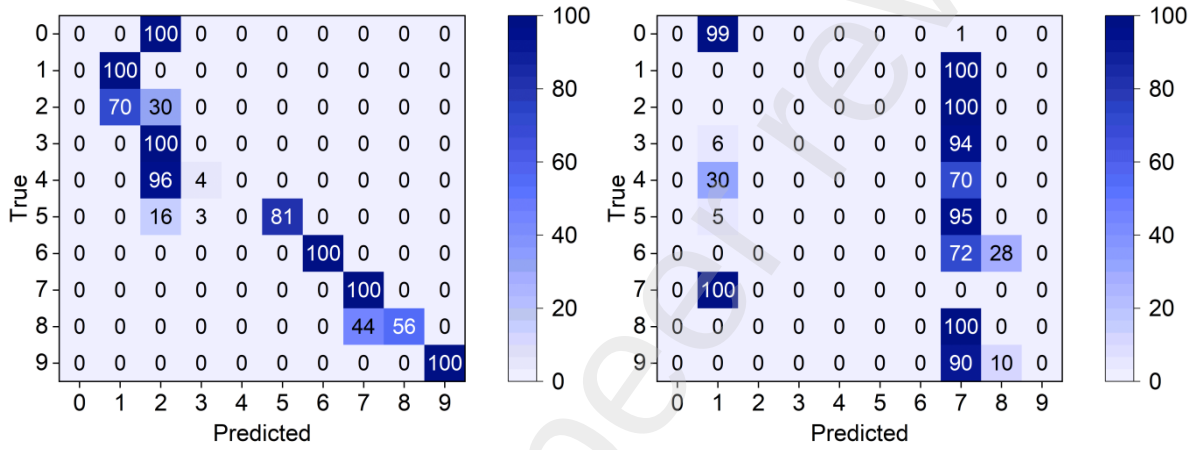
Fig. 11 Adversarial results with 10 PFs: (a) frequency response, (b) model prediction.

Reducing the number of PFs, through trials, shows that a nontarget attack can be achieved with alterations in as few as 2 PFs, with the adversarial frequency change shown in **Fig. 12a**. Nontarget attacks on a minimal number of PFs tend to perturb the prediction to the label closest to the original (e.g., DC 4), as shown in **Fig. 12b**. Such adversarial modifications are even harder to detect. These findings highlight the vulnerability of data-driven SHM systems. The success rate of nontarget attacks is defined as the percentage of adversarial samples that have been successfully identified as an arbitrary target class. When using 10 PFs and considering all DCs, their confusion matrix is presented in **Fig. 13a**, with an overall success rate of 43.3%. It can be observed that although the bridge state DC 0 can be effectively disturbed to other DCs, some DCs remain robust against such attacks. As there is a clear difference in features between them; changing only a few features is insufficient to bridge this difference in the feature space. However, simply by increasing the number of PFs, that is, the dimensionality of the perturbation, this robustness can be easily broken, as shown in **Fig. 13b**. When using 100 PFs (4% of the total number of features), the success rate increases to 100%. In fact, the number of PFs can be further reduced, but this is not the focus of this work. So far, from a data science perspective, the SHM system has been hacked by the PFA algorithm. However, in practical engineering, it is a challenge to write these adversarial alterations into system samples; outside attackers usually lack such authorities. This leads to the concept of the hacker vehicle employed as a tool for editing bridge signals.



(a) (b)

Fig. 12 Adversarial results with 2 PFs: (a) frequency response, (b) model prediction.



(a) (b)

Fig. 13 Confusion matrix for adversarial samples: (a) 10 PFs, (b) 100 PFs.

4.1.2 Demonstration of hacker vehicle

We install an exciter on vehicle#0 to construct a hacker vehicle, indirectly editing the bridge vibration signals through the VBI process. The exciter is controlled by Equation (19). As shown in **Fig. 14**, experimental trials reveal a roughly linear relationship between the bridge amplitudes for the top ten PFs and the exciter amplitude $\boldsymbol{A}$.

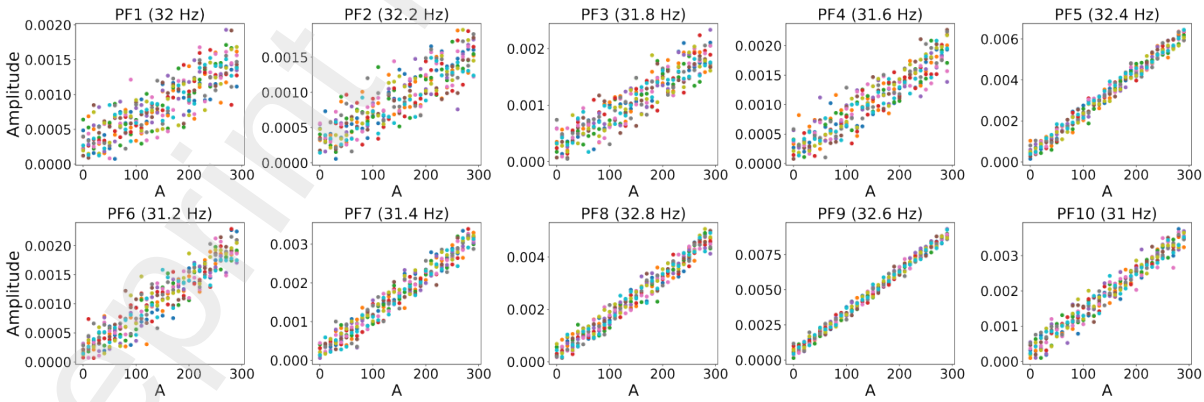


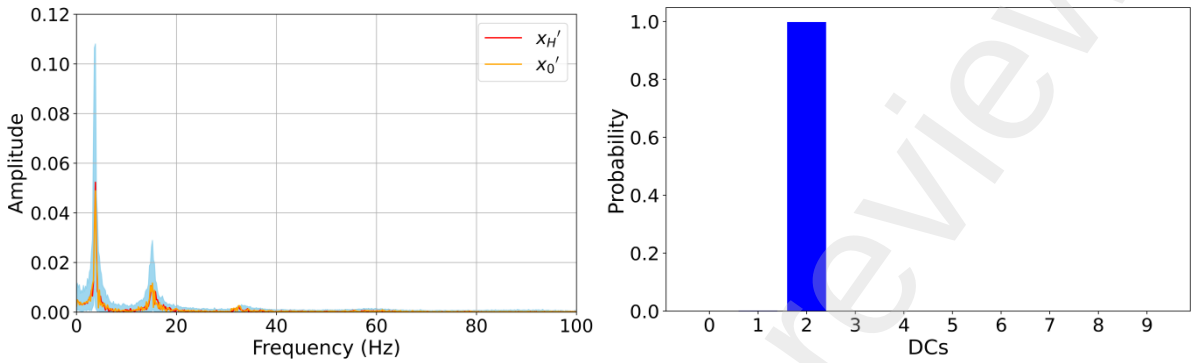
Fig. 14 Bridge amplitudes at PFs corresponding to the exciter amplitude A.

Based on Equations (17) to (21) and the output $\boldsymbol{x_{opt}}$ from **Algorithm 2**, it can be determined that:

$$\boldsymbol{\kappa} = \{32, 31.6, 32.4, 31.4, 32.8, 32.6, 31\} \#(22)$$

$$\boldsymbol{A} = \{64.1360, 20.1116, 39.8875, 22.1657, 34.5457, 21.7874, 8.3988\} \#(23)$$

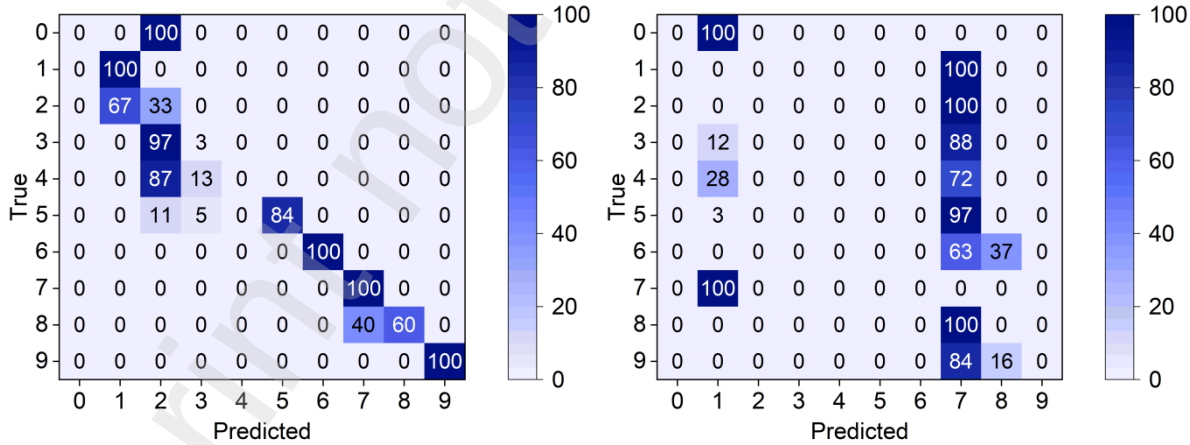
They can be used to construct the exciter function $\boldsymbol{\Gamma = Acos(2\pi\kappa t)}$. The hacker vehicle equipped with this exciter drives on the bridge, by which a sample $\boldsymbol{x_H{}'}$ (already pre-processed) can be collected via the bridge sensor. **Fig. 15a** compares the sample $\boldsymbol{x_H{}'}$ and the adversarial sample $\boldsymbol{x_0{}'}$ produced by the PFA algorithm, which are very similar. **Fig. 15b** shows that its prediction is consistent with the above adversarial result that the model misclassifies healthy samples DC 0 as DC 2 with nearly 100% probability. These demonstrate that the hacker vehicle can effectively write adversarial alterations into the system without any authority.



(a) (b)

Fig. 15 Sample from the hacker vehicle: (a) frequency response, (b) model prediction.

When using the hacker vehicle to reconstruct adversarial alterations in 10 PFs and 100 PFs, and considering all DCs, their confusion matrices are shown in **Fig. 16a** and **Fig. 16b**, respectively. Their patterns are consistent with the PFA results above, with success rates of 42% and 100%, respectively. This also proves the effectiveness of the hacker vehicle in case of nontarget attack. This type of nontarget attack is very close to a normal event visually and does not deviate from the physical response to it. With the introduction of the hacker vehicle, adversarial changes can be effectively written into the system without any authorization; defenses in data security are not even effective against it.



(a) (b)

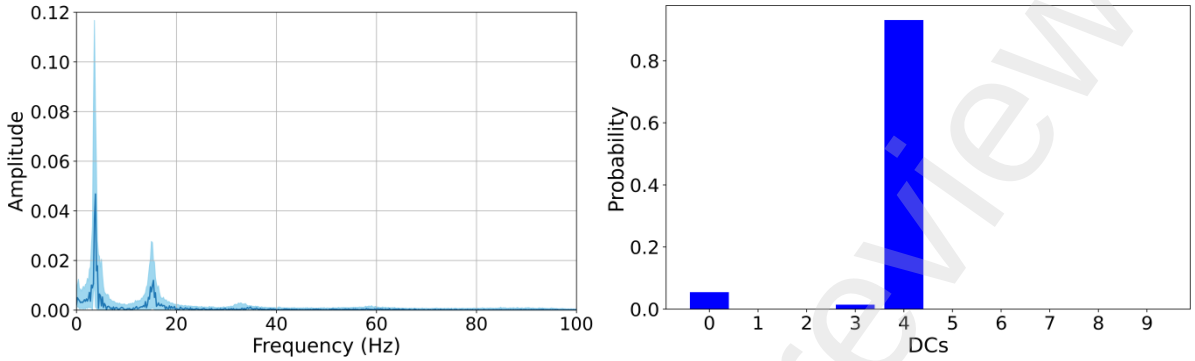
Fig. 16 Confusion matrix for samples from the hacker vehicle: (a) 10 PFs, (b) 100 PFs.

4.2. Target attack

4.2.1 Demonstration of PFA algorithm

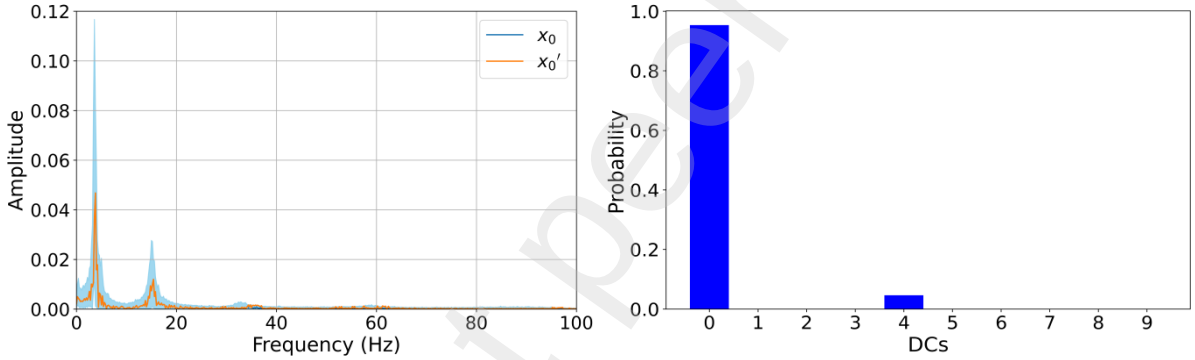
A typical target attack aims to fool the SHM system to diagnose a damaged bridge as healthy, which could be riskier than a nontarget attack. DC 4 will be first used as the attack target to demonstrate how to fool the model to classify it as DC 0 (healthy). Target attacks generally require more PFs than nontarget attacks. We select 100 PFs, and their feature impacts can be referenced in **Fig. 9**. Based on the $\boldsymbol{V^0}$ event as input, the bridge's frequency response and model predictions are shown in **Fig. 17a**

and **Fig. 17b**, respectively. Model $\mathcal{M}$ predicts the bridge condition as DC 4 with a 93.1% probability, which is damage on 5-th element, μ=0.9. This could represent a moderate damage in practical engineering. However, for the adversarial results of the target attack using **Algorithm 3**, despite its frequency response being visually indistinguishable from the original, the model predicts it as DC 0 (healthy) with a 95.4% probability, as shown in **Fig. 18a** and **Fig. 18b**. This indicates the target attack has worked well.



(a) (b)

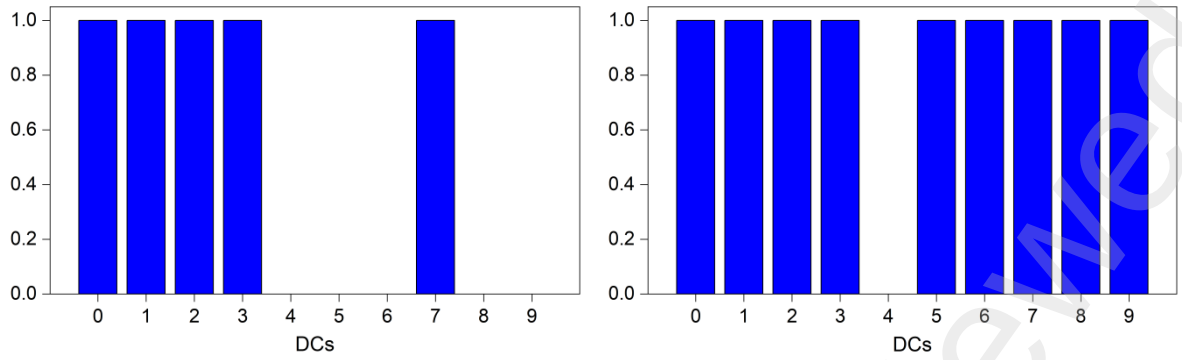
Fig. 17 Original results: (a) frequency response, (b) model prediction.



(a) (b)

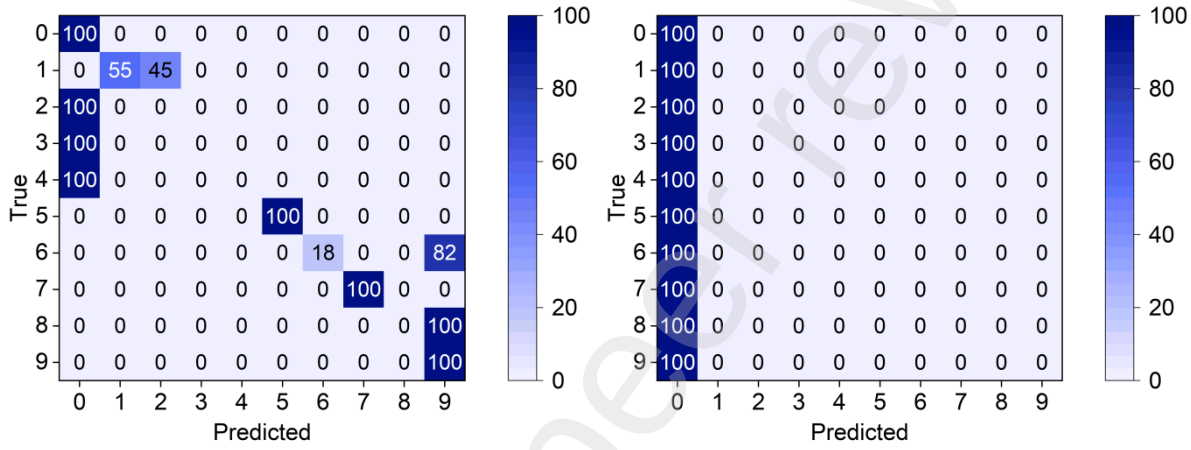
Fig. 18 Target attack results: (a) frequency response, (b) model prediction.

After that, attempts are made to perturb DC 4 into all other DCs. **Fig. 19a** and **Fig. 19b** show the percentage of samples (totally 100 samples) where DC 4 has been successfully perturbed to the target labels using 100 and 200 PFs, respectively. It appears that target attacks are more effective against DCs of damage severities. This is because, in the feature space, DC 4 is closer to DCs of different damage severities than to DCs of different damage locations, allowing it to be perturbed to other DCs with fewer feature changes. Nevertheless, this can be addressed by increasing the number of PFs; for example, using 200 PFs can perturb DC 4 to any DCs with a 100% success rate. The success rate of target attacks is defined as the percentage of adversarial samples that have been successfully identified as the target class. When perturbing all DCs to DC 0, their confusion matrices, as shown in **Fig. 20a** and **Fig. 20b**, indicate an overall success rate of 40% (100 PFs) and 100% (200 PFs). It is found that DCs with a large difference to DC 0 (e.g., different damage location and severe damage) remain robustness against such attacks when there are few PFs. Increasing the PFs undermines this robustness, achieving a 100% success rate in fooling the system. These indicate that the adversarial alterations obtained through the PFA algorithm are effective. Then, the hacker vehicle will attempt to reproduce these modifications and illegally write them into the system.



(a) (b)

Fig. 19 Perturbation from DC 4 to others: (a) 100 PFs, (b) 200 PFs.



(a) (b)

Fig. 20 Confusion matrix for target attacks: (a) 100 PFs, (b) 200 PFs.

4.2.2 Demonstration of hacker vehicle

For the target attack perturbing DC 4 to DC 0, with 100 PFs, the configuration of the exciter is shown in **Fig. 21**, where the impact of each feature and the corresponding exciter amplitude can be obtained. **Fig. 22a** compares the results from the hacker vehicle equipped with the exciter, $\boldsymbol{x_H}'$, to the adversarial samples generated by the PFA algorithm, $\boldsymbol{x_0}'$. It is evident that the two are very similar, indicating that the hacker vehicle can well reconstruct the adversarial alterations of the target attack. Moreover, **Fig. 22b** shows that prediction from the hacker vehicle is consistent with the PFA results. Considering all DCs, their confusion matrices are displayed in **Fig. 23a** and **Fig. 22b**, respectively. They are also consistent with the PFA matrices, with success rates of 39.5% and 100%, respectively. These results demonstrate the effectiveness of the hacker vehicle in target attacks. Furthermore, these malicious samples are difficult to detect visually or physically and cannot be guarded by data security strategies.

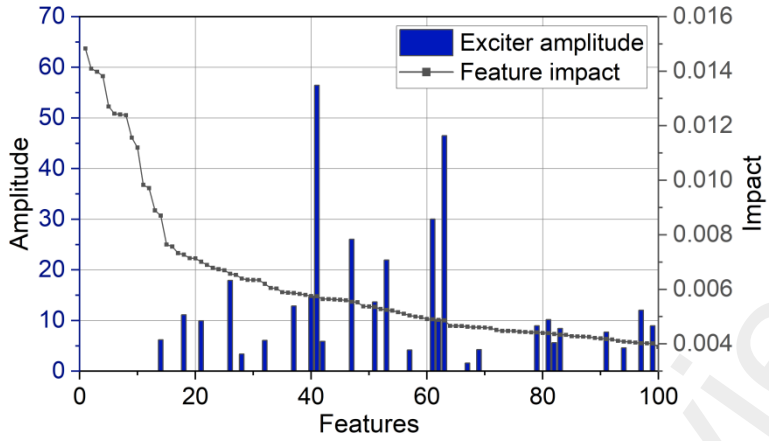
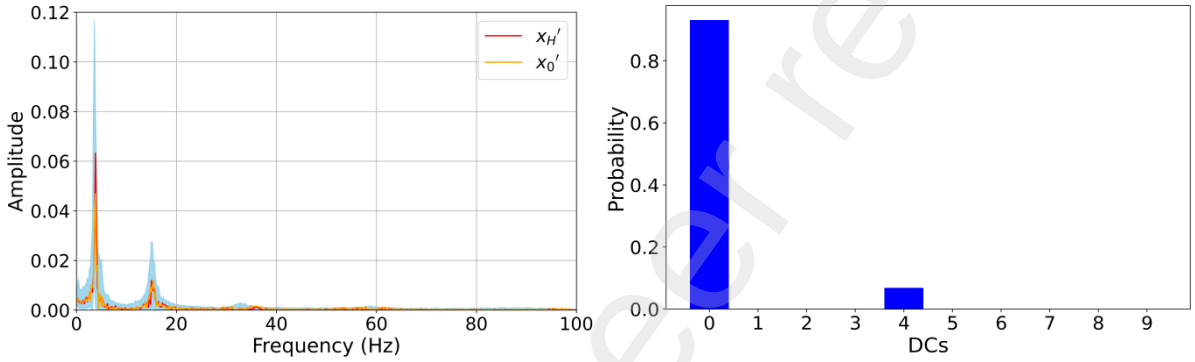
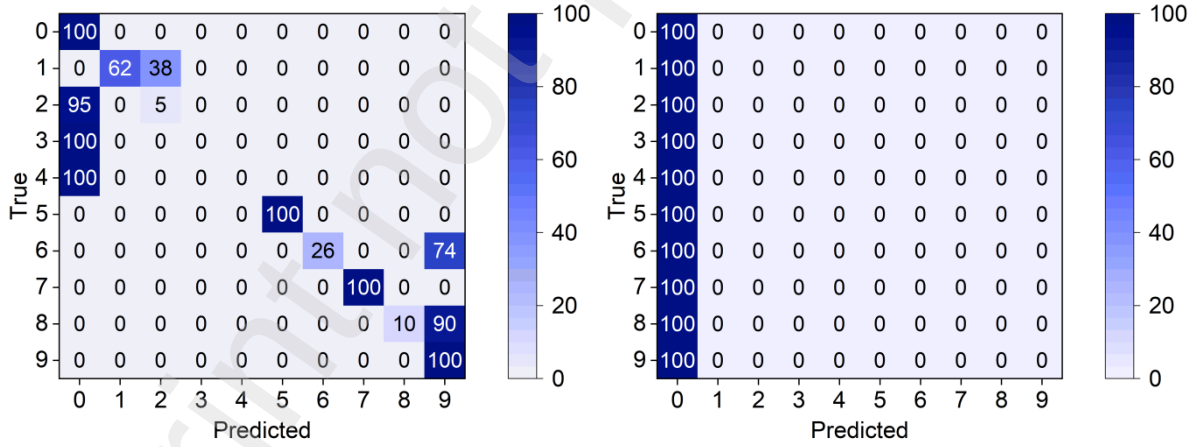


Fig. 21 Configuration of the exciter.



(a) (b)

Fig. 22 Sample from the hacker vehicle: (a) frequency response, (b) model prediction.



(a) (b)

Fig. 23 Confusion matrix for samples from the hacker vehicle: (a) 100 PFs, (b) 200 PFs.

4.3. Feasibility on different models

In fact, this attack method is effective on different models. In addition to model $\mathcal{M}$, we trained two other commonly used networks on the original dataset; they are Network in Network (NiN) and VGG-16. NiN is a DL network architecture that inserts additional 1×1 convolutional layers inside traditional convolutional layers to enhance the model's representational power [40]. VGG-16 is a popular CNN architecture, and the significance of it lies in its simple yet effective architecture that demonstrates excellent performance on classification tasks [41]. The network configurations have been kept similar to the original, with some modifications to adapt to the SHM task of this study.

Their details are shown in **Table 3** and **Table 4**, respectively. It should be noted that since NiN and VGG-16 are originally designed for complex, large-scale tasks, applying them to the relatively small dataset of this study may lead to overfitting issues. To address these, dropout layers and an early stopping strategy have been added to the networks. The dropout rate is 0.3, and the early stopping condition is that no improvement in validation loss is observed within 20 epochs.

Table 3 NiN configurations

Layer	Output shape	Parameter	Activation
Conv1d	2500 × 64	Kernel number: 64; Kernel size:10; Stride: 1; Padding: "same"	LeakyReLU
Conv1d	2500 × 64	Kernel number: 64; Kernel size:1; Stride: 1; Padding: "same"	LeakyReLU
Max pooling	1250 × 64	Kernel: 2; Stride: 2	None
Dropout	1250 × 64	Rate: 0.3	None
Conv1d	1250 × 128	Kernel number: 128; Kernel size:10; Stride: 1; Padding: "same"	LeakyReLU
Conv1d	1250 × 128	Kernel number: 128; Kernel size:1; Stride: 1; Padding: "same"	LeakyReLU
Max pooling	625 × 128	Kernel: 2; Stride: 2	None
Dropout	625 × 128	Rate: 0.3	None
Conv1d	625 × 30	Kernel number: 30; Kernel size:1; Stride: 1; Padding: "same"	LeakyReLU
Flatten	18750	None	None
Dense	10	None	Softmax

Table 4 VGG-16 configurations

Layer	Output shape	Parameter	Activation
Conv1d	2500 × 32	Kernel number: 32; Kernel size:10; Stride: 1; Padding: "same"	LeakyReLU
Max pooling	1250 × 32	Kernel: 2; Stride: 2	None
Conv1d	1250 × 64	Kernel number: 64; Kernel size:10; Stride: 1; Padding: "same"	LeakyReLU
Max pooling	625 × 64	Kernel: 2; Stride: 2	None
Conv1d	625 × 128	Kernel number: 128; Kernel size:10; Stride: 1; Padding: "same"	LeakyReLU
Max pooling	312 × 128	Kernel: 2; Stride: 2	None
Conv1d	312 × 128	Kernel number: 128; Kernel size:10; Stride: 1; Padding: "same"	LeakyReLU
Max pooling	156 × 128	Kernel: 2; Stride: 2	None
Dropout	156 × 128	Rate: 0.3	None
Flatten	19968	None	None
Dense	100	None	None
Dense	10	None	Softmax

The model training has been carried out within the environment and settings of Section 2.3. The training histories of NiN and VGG-16 are depicted in **Fig. 24a** and **Fig. 24b**, respectively. They ultimately achieved test accuracies of 91.6% (NiN) and 92.8% (VGG-16), respectively, which are slightly higher than model $\mathcal{M}$'s accuracy of 91.2%. Both models stabilize before reaching 100 epochs, and according to their trends in test loss, there are no overfitting issues.

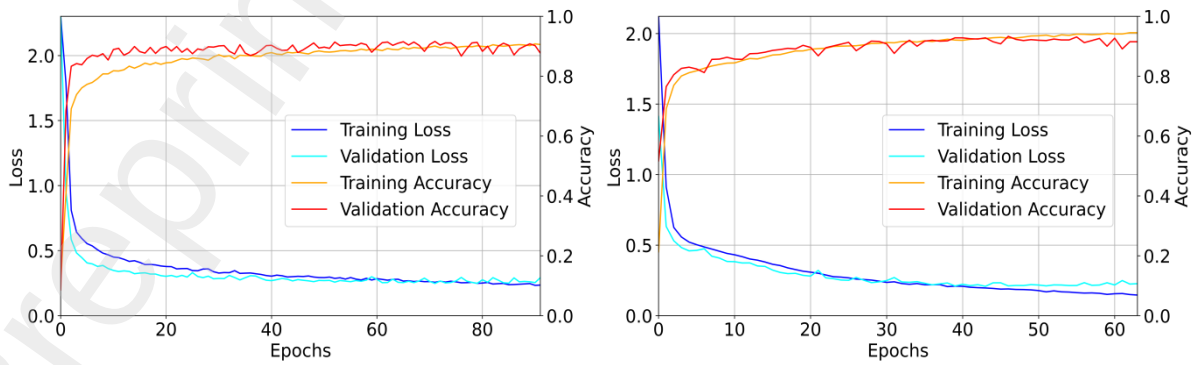
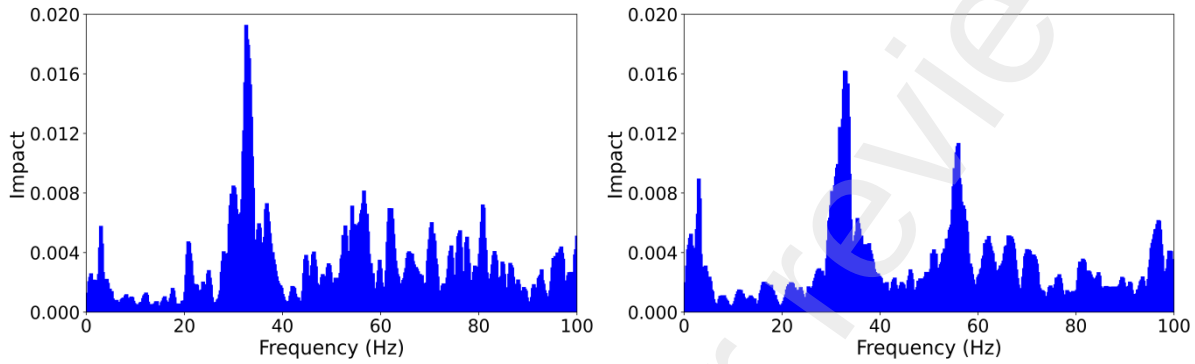


Fig. 24 Training histories for different models: (a) NiN, (b) VGG-16.

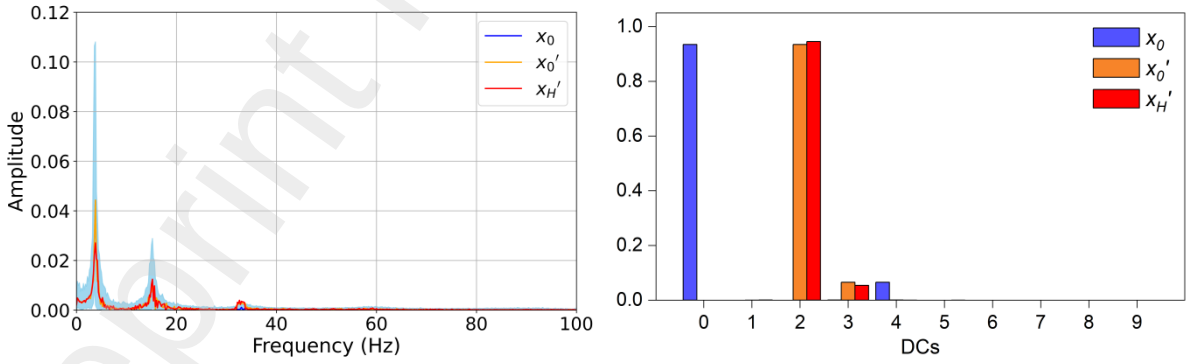
Eavesdropping on samples from DC 0 (healthy) and adopting **Algorithm 1**, the feature impacts of these models are illustrated in **Fig. 25a** and **Fig. 25b**, where only features within 0-100 Hz are shown as other features have minor impacts. Clearly, some features near the first four frequencies of the bridge indeed receive more attention (3.8, 15.2, 34.2, and 60.8 Hz); however, the weights assigned to them are different, not to mention some features that are given high weights but whose physical significance remains unclear. This encourages the studies on model interpretability and trustworthiness. When selecting models, one may consider more than just the accuracy of results. Anyway, the main focus here is to demonstrate that for these models, they are all vulnerable to adversarial attacks.



(a) (b)

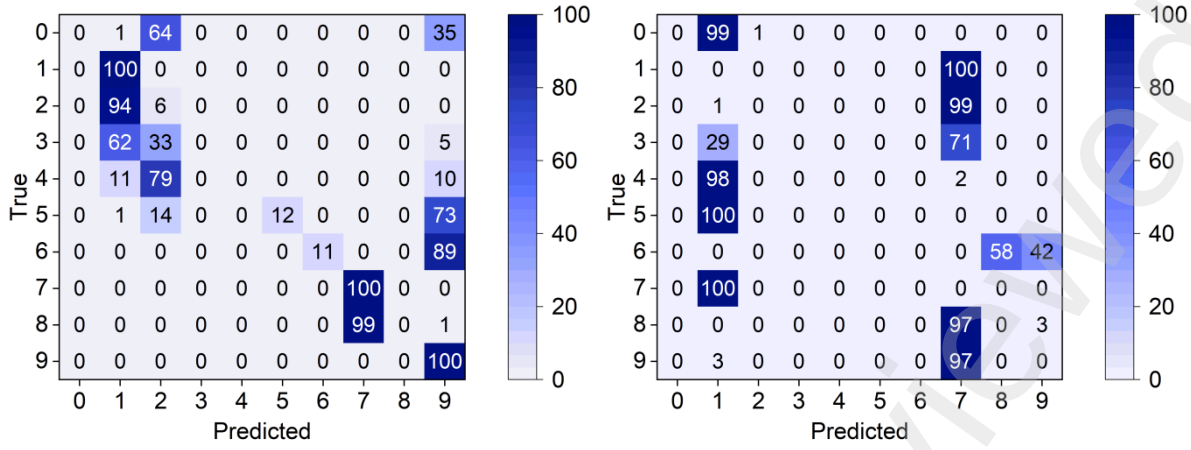
Fig. 25 Feature impacts: (a) NiN, (b) VGG-16.

Taking the nontarget attack on the NiN model with 10 PFs of DC 0 as a case study, **Fig. 26a** and **Fig. 26b** compare the frequency domain responses and model predictions for $\boldsymbol{x_0}$, $\boldsymbol{x_0}'$, and $\boldsymbol{x_H}'$. The proposed PFA algorithm successfully perturbs the model's predictions with inconspicuous PF modifications, and the hacker vehicle effectively reproduces these modifications. Originally, the model predicted a 93.4% probability that a sample belonged to DC 0, but in the adversarial and hacker vehicle results, the model's predictions shifted to a 93.5% and 94.5% probability of belonging to DC 2, respectively. **Fig. 27a** displays the confusion matrices for nontarget attacks on NiN using 10 and 100 PFs across all DCs, with overall success rates of 67.1% and 100%. Similarly, for target attacks aimed at perturbing samples to DC 0, the confusion matrices for NiN using 100 and 200 PFs across all DCs, shown in **Fig. 28**, have overall success rates of 48.8% and 100%. Due to space limitations, only the results for the hacker vehicle are presented here, but it can be known from the analysis that they closely replicate the PFA attacks. Clearly, the attack on NiN has been successful.



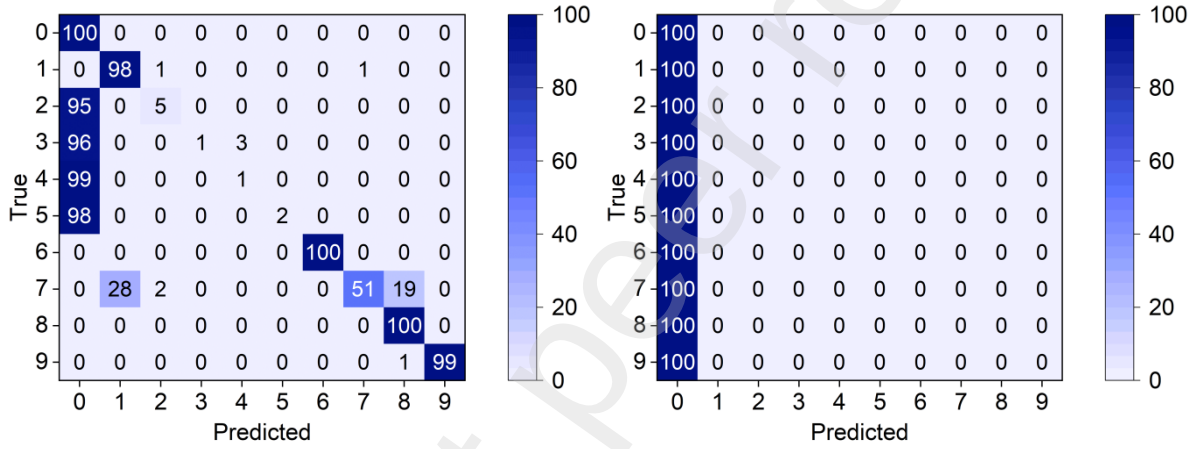
(a) (b)

Fig. 26 Attack on NiN: (a) frequency response, (b) model prediction.



(a) (b)

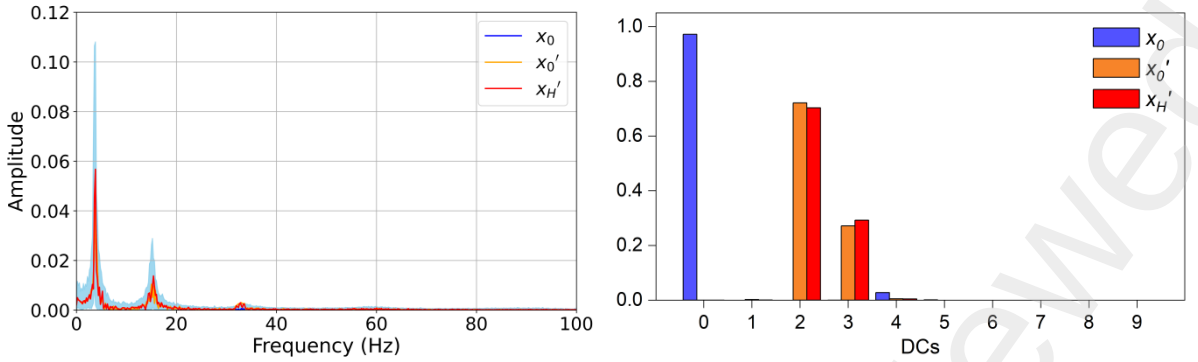
Fig. 27 Confusion matrix for nontarget attack: (a) 10 PFs, (b) 100 PFs.



(a) (b)

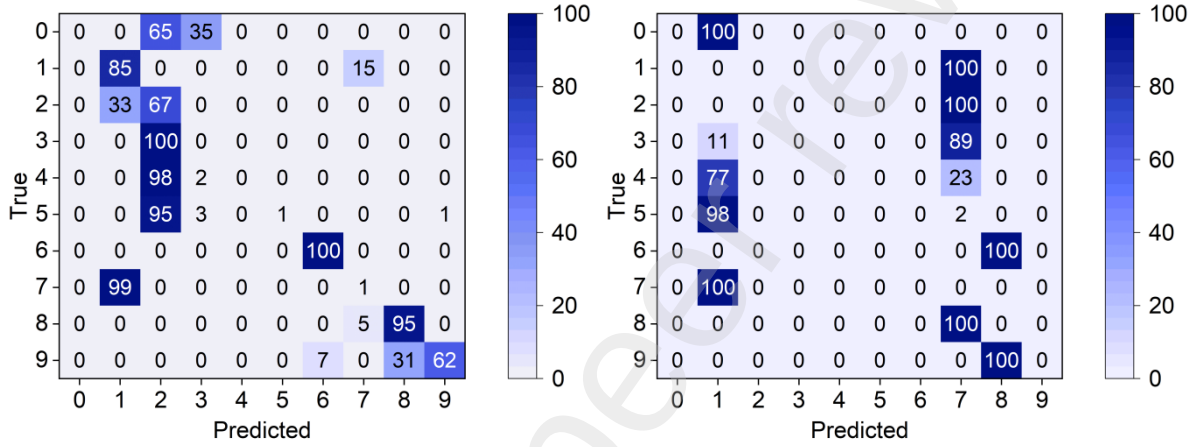
Fig. 28 Confusion matrix for target attack: (a) 100 PFs, (b) 200 PFs.

Fig. 29a and **Fig. 29b** compare the frequency domain responses and model predictions for $\boldsymbol{x_0}$, $\boldsymbol{x_0}'$, and $\boldsymbol{x_H}'$ of the VGG-16 model. Initially, the model predicted with a 97.2% probability that a sample belonged to DC 0, but after adversarial alterations, the predictions were disturbed to probabilities of 72.1% and 70.3% of belonging to DC 2, respectively; the results from the hacker vehicle were similar to those from the PFA. In the confusion matrices for the hacker vehicle's nontarget attacks on the VGG-16 model across all DCs, utilizing 10 and 100 PFs (see **Fig. 30**), the overall success rates are 58.9% and 100%, respectively. For the target attack that perturbs samples to DC 0, the confusion matrices using 100 and 200 PFs can be referred to **Fig. 31**. Their overall success rates are 48.7% and 100%. These results demonstrate the success of the proposed method in fooling ML models. Furthermore, one may observe that both nontarget and target attacks on NiN and VGG-16 models seem to have higher success rates, despite their structures appearing more complex than model $\mathcal{M}$ (their accuracy also being a bit higher than model $\mathcal{M}$). This suggests that models with greater complexity may be more "vulnerable" to malicious attacks, and considerations should extend beyond mere accuracy when choosing models. However, such vulnerabilities might be mitigated by defensive strategies. In any case, this is a topic worthy of discussion.



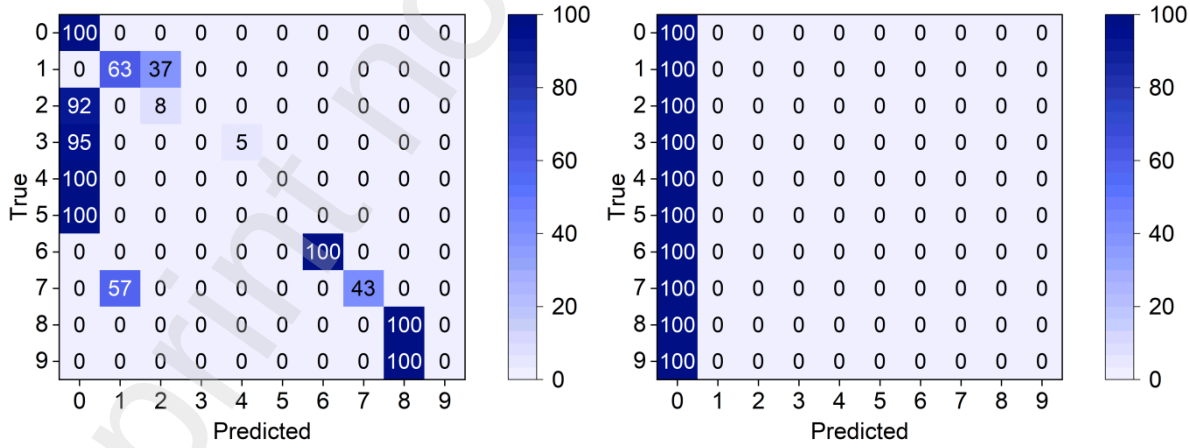
(a) (b)

Fig. 29 Attack on VGG-16: (a) frequency response, (b) model prediction.



(a) (b)

Fig. 30 Confusion matrix for nontarget attack: (a) 10 PFs, (b) 100 PFs.



(a) (b)

Fig. 31 Confusion matrix for target attack: (a) 100 PFs, (b) 200 PFs.

5. Conclusion

This paper argues that data-driven SHM frameworks are vulnerable and demonstrates how to hack a DL-based bridge SHM system from an attacker's perspective. This raises concerns about model

interpretability and trustworthiness, rather than just prediction accuracy. But more than that, it gives meaning to a new field - the defence of SHM systems. The paper first proposes a PFA algorithm that seeks moderate alterations to the most influential features within physical constraints to attack the model, which is a black-box attack algorithm. It then conceptualizes a hacker vehicle that, by passing over the bridge, indirectly edits the bridge vibration signals through the VBI process, thus physically writing adversarial alterations into the system. Furthermore, through a case of a typical traffic event-based bridge SHM, the method's performance is evaluated. Based on the results, the following conclusions can be drawn:

1. The proposed PFA algorithm can effectively identify the PFs that most significantly impact the model results, then seek moderate alterations to disturb model predictions to the desired outcomes. These modifications are visually similar to the original and comply with physical constraints, making them hard to detect.
2. The hacker vehicle can effectively reproduce the PFA algorithm's alterations and unauthorizedly write them into system samples. The model's predictions for samples from the hacker vehicle closely resemble those for the PFA samples. This physical attack extends defence work beyond the discipline of data security.
3. In nontarget attacks, altering as few as 2 PFs can disturb a structural state from healthy (DC 0) to damaged (DC 4). For some robust cases, increasing the number of PFs can improve the success rate of the attack. For all cases, it can attack the model with a 100% success rate by disturbing 100 PFs.
4. In target attacks, the selected structural state can be disturbed to any desired state; increasing PFs can also increase its success rate. For all cases, it can attack the model with a 100% success rate by disturbing 200 PFs.
5. By attacking different models, the method's general effectiveness is proven. Interestingly, complex models appear to be more vulnerable to the attacks.

Future studies will involve configuring a real hacker vehicle for experimentation, the key to which is the adaptive exciter as described in the text; this could be made with a smartphone and a tailor-made program on it. Importantly, how to defend against such threats is a field for future exploration. We hope this paper will attract valuable ideas and discussions.

Acknowledgement

This research is sponsored by the Jane and Aatos Erkko Foundation in Finland (Grant No. 210018).

References

[1] Infrastructure of America, A comprehensive assessment of America's infrastructure, ASCE, 2021.
[2] K. Gkoumas, F. Marques dos Santos, M. van Balen, A. Tsakalidis, A. Ortega, M. Grosso, A. Haq, F. Pekár, Research and innovation in bridge maintenance, inspection and monitoring A European perspective based on the Transport Research and Innovation Monitoring and Information System (TRIMIS), 2019. https://doi.org/10.2760/16174.
[3] C.-Z. Dong, F.N. Catbas, A review of computer vision–based structural health monitoring at local and global levels, Structural Health Monitoring 20 (2021) 692–743. https://doi.org/10.1177/1475921720935585.
[4] L. Hui, O. Jinping, Structural Health Monitoring: From Sensing Technology Stepping to Health Diagnosis, Procedia Engineering 14 (2011) 753–760. https://doi.org/10.1016/j.proeng.2011.07.095.
[5] S. Hassani, U. Dackermann, A Systematic Review of Optimization Algorithms for Structural Health Monitoring and Optimal Sensor Placement, Sensors 23 (2023) 3293. https://doi.org/10.3390/s23063293.
[6] E.P. Carden, P. Fanning, Vibration Based Condition Monitoring: A Review, Structural Health Monitoring 3 (2004) 355–377. https://doi.org/10.1177/1475921704047500.

[7] R. Du, P. Santi, M. Xiao, A. V. Vasilakos, C. Fischione, The Sensable City: A Survey on the Deployment and Management for Smart City Monitoring, IEEE Communications Surveys & Tutorials 21 (2019) 1533–1560. https://doi.org/10.1109/COMST.2018.2881008.
[8] O. Avci, O. Abdeljaber, S. Kiranyaz, M. Hussein, M. Gabbouj, D.J. Inman, A review of vibration-based damage detection in civil structures: From traditional methods to Machine Learning and Deep Learning applications, Mechanical Systems and Signal Processing 147 (2021) 107077. https://doi.org/10.1016/j.ymssp.2020.107077.
[9] L. Sun, Z. Shang, Y. Xia, S. Bhowmick, S. Nagarajaiah, Review of Bridge Structural Health Monitoring Aided by Big Data and Artificial Intelligence: From Condition Assessment to Damage Detection, Journal of Structural Engineering 146 (2020) 04020073. https://doi.org/10.1061/(ASCE)ST.1943-541X.0002535.
[10] M.M. Alamdari, T. Rakotoarivelo, N.L.D. Khoa, A spectral-based clustering for structural health monitoring of the Sydney Harbour Bridge, Mechanical Systems and Signal Processing 87 (2017) 384–400. https://doi.org/10.1016/j.ymssp.2016.10.033.
[11] M.H. Rafiei, H. Adeli, A novel unsupervised deep learning model for global and local health condition assessment of structures, Engineering Structures 156 (2018) 598–607. https://doi.org/10.1016/j.engstruct.2017.10.070.
[12] Y. Zhang, Y. Miyamori, S. Mikami, T. Saito, Vibration-based structural state identification by a 1-dimensional convolutional neural network, Computer-Aided Civil and Infrastructure Engineering 34 (2019) 822–839. https://doi.org/10.1111/mice.12447.
[13] M.Z. Sarwar, D. Cantero, Deep autoencoder architecture for bridge damage assessment using responses from several vehicles, Engineering Structures 246 (2021) 113064. https://doi.org/10.1016/j.engstruct.2021.113064.
[14] Y. Lan, Y. Zhang, W. Lin, Diagnosis algorithms for indirect bridge health monitoring via an optimized AdaBoost-linear SVM, Engineering Structures 275 (2023) 115239. https://doi.org/10.1016/j.engstruct.2022.115239.
[15] Y. Lan, Z. Li, W. Lin, A Time-Domain Signal Processing Algorithm for Data-Driven Drive-by Inspection Methods: An Experimental Study, Materials 16 (2023) 2624. https://doi.org/10.3390/ma16072624.
[16] Z. Kong, J. Xue, Y. Wang, L. Huang, Z. Niu, F. Li, A Survey on Adversarial Attack in the Age of Artificial Intelligence, Wireless Communications and Mobile Computing 2021 (2021) e4907754. https://doi.org/10.1155/2021/4907754.
[17] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song, Robust Physical-World Attacks on Deep Learning Visual Classification, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018: pp. 1625–1634. https://doi.org/10.1109/CVPR.2018.00175.
[18] I. Fursov, M. Morozov, N. Kaploukhaya, E. Kovtun, R. Rivera-Castro, G. Gusev, D. Babaev, I. Kireev, A. Zaytsev, E. Burnaev, Adversarial Attacks on Deep Models for Financial Transaction Records, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, New York, NY, USA, 2021: pp. 2868–2878. https://doi.org/10.1145/3447548.3467145.
[19] M.D. Champneys, A. Green, J. Morales, M. Silva, D. Mascarenas, On the vulnerability of data-driven structural health monitoring models to adversarial attack, Structural Health Monitoring 20 (2021) 1476–1493. https://doi.org/10.1177/1475921720920233.
[20] B. Finley, E.A. Schneider, ICAS: the center of diagnostics and prognostics for the United States Navy, in: Component and Systems Diagnostics, Prognosis, and Health Management, SPIE, 2001: pp. 186–193. https://doi.org/10.1117/12.434237.
[21] S. Qiu, Q. Liu, S. Zhou, C. Wu, Review of Artificial Intelligence Adversarial Attack and Defense Technologies, Applied Sciences 9 (2019) 909. https://doi.org/10.3390/app9050909.
[22] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z.B. Celik, A. Swami, Practical Black-Box Attacks against Machine Learning, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, Association for Computing Machinery, New York, NY, USA, 2017: pp. 506–519. https://doi.org/10.1145/3052973.3053009.

[23] A. Diez, N.L.D. Khoa, M. Makki Alamdari, Y. Wang, F. Chen, P. Runcie, A clustering approach for structural health monitoring on bridges, Journal of Civil Structural Health Monitoring 6 (2016) 429–445. https://doi.org/10.1007/s13349-016-0160-0.
[24] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D.J. Inman, 1D convolutional neural networks and applications: A survey, Mechanical Systems and Signal Processing 151 (2021) 107398. https://doi.org/10.1016/j.ymssp.2020.107398.
[25] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[26] C. Liu, Y. Zhu, H. Ye, Bridge frequency identification based on relative displacement of axle and contact point using tire pressure monitoring, Mechanical Systems and Signal Processing 183 (2023) 109613. https://doi.org/10.1016/j.ymssp.2022.109613.
[27] Y. Lan, Z. Li, K. Koski, L. Fülöp, T. Tirkkonen, W. Lin, Bridge frequency identification in city bus monitoring: A coherence-PPI algorithm, Engineering Structures 296 (2023) 116913. https://doi.org/10.1016/j.engstruct.2023.116913.
[28] R. Corbally, A. Malekjafarian, A deep-learning framework for classifying the type, location, and severity of bridge damage using drive-by measurements, Computer-Aided Civil and Infrastructure Engineering 00 (2023) 1–20. https://doi.org/10.1111/mice.13104.
[29] Y. Lan, Z. Li, W. Lin, Physics-guided diagnosis framework for bridge health monitoring using raw vehicle accelerations, Mechanical Systems and Signal Processing 206 (2024) 110899. https://doi.org/10.1016/j.ymssp.2023.110899.
[30] P. Múčka, Simulated Road Profiles According to ISO 8608 in Vibration Analysis, Journal of Testing and Evaluation 46 (2018) 20160265. https://doi.org/10.1520/JTE20160265.
[31] H. Xu, Y.H. Liu, Z.L. Wang, K. Shi, B. Zhang, Y.B. Yang, General contact response of single-axle two-mass test vehicles for scanning bridge frequencies considering suspension effect, Engineering Structures 270 (2022) 114880. https://doi.org/10.1016/j.engstruct.2022.114880.
[32] Z. Li, Y. Lan, W. Lin, Indirect damage detection for bridges using sensing and temporarily parked vehicles, Engineering Structures 291 (2023) 116459. https://doi.org/10.1016/j.engstruct.2023.116459.
[33] J.K. Sinha, M.I. Friswell, S. Edwards, Simplified models for the location of cracks in beam structures using measured vibration data, Journal of Sound and Vibration 251 (2002) 13–38. https://doi.org/10.1006/jsvi.2001.3978.
[34] W.M. Ostachowicz, M. Krawczuk, On Modelling of Structural Stiffness Loss Due to Damage, Key Engineering Materials 204–205 (2001) 185–200. https://doi.org/10.4028/www.scientific.net/KEM.204-205.185.
[35] B. Xu, N. Wang, T. Chen, M. Li, Empirical Evaluation of Rectified Activations in Convolutional Network, (2015). https://doi.org/10.48550/arXiv.1505.00853.
[36] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, TensorFlow: A system for large-scale machine learning, (2016). https://doi.org/10.48550/arXiv.1605.08695.
[37] T. Hastie, R. Tibshirani, J.H. Friedman, The elements of statistical learning: data mining, inference, and prediction, 2nd ed, Springer, New York, 2009.
[38] J. Su, D.V. Vargas, S. Kouichi, One pixel attack for fooling deep neural networks, IEEE Trans. Evol. Computat. 23 (2019) 828–841. https://doi.org/10.1109/TEVC.2019.2890858.
[39] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell 1 (2007) 33–57. https://doi.org/10.1007/s11721-007-0002-0.
[40] M. Lin, Q. Chen, S. Yan, Network In Network, (2014). https://doi.org/10.48550/arXiv.1312.4400.
[41] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, (2015). https://doi.org/10.48550/arXiv.1409.1556.