

Development of an automatic and knowledge-infused framework for structural health monitoring based on prompt engineering

Truong-Thang NGUYEN^a, Viet-Hung DANG^{a,b*}, Thanh-Tung PHAM^{a,b}

^a Faculty of Building and Industrial Construction, Hanoi University of Civil Engineering, Hanoi 11600, Vietnam

^b Research Group of Development and application of advanced materials and modern technologies in construction, Hanoi University of Civil Engineering, Hanoi 11600, Vietnam

*Corresponding author. E-mail: hungdv@huce.edu.vn

© Higher Education Press 2024

ABSTRACT Collecting and analyzing vibration signals from structures under time-varying excitations is a non-destructive structural health monitoring approach that can provide meaningful information about the structures' safety without interrupting their normal operations. This paper develops a novel framework using prompt engineering for seamlessly integrating users' domain knowledge about vibration signals with the advanced inference ability of well-trained large language models (LLMs) to accurately identify the actual states of structures. The proposed framework involves formulating collected data into a standardized form, utilizing various prompts to gain useful insights into the dynamic characteristics of vibration signals, and implementing an in-house program with the help of LLMs to perform damage detection. The advantages, as well as limitations, of the proposed method are qualitatively and quantitatively assessed through two realistic case studies from literature, demonstrating that the present method is a new way to quickly construct practical and reliable structural health monitoring applications without requiring advanced programming/mathematical skills or obscure specialized programs.

KEYWORDS structural health monitoring, vibration, large language model, signal processing, prompt engineering

1 Introduction

This study aims to make a step toward building automated structural health monitoring (SHM) applications for civil structures that can intelligently monitor the actual operational states of structures and proactively provide engineers and owners with damage-related information without requiring them to invest excessive time and effort in programming. SHM is a sub-field of civil engineering that aims to monitor the actual operational states of structures and provide engineers and owners with critical damage-related information such as damage occurrence, damage location and damage extent. This information helps engineers proactively devise maintenance actions to ensure the structures' safety and extend their service life. Besides expertise in structural analysis, SHM requires

knowledge from diverse domains, including signal vibrations, data analytics, and programming skills to effectively manage large volume of data gathered from full-scale structures. For instance, an apparent indicator of damage occurrence is when the amplitudes of structures' responses, such as deflection, deformation, settlement... exceed predefined safety thresholds. Thus, various statistical properties in the time domain can be used to perform SHM, like crossing rates, mean values, number of peaks, etc., as demonstrated in Ref. [1]. In practice, temporal signals are contaminated by different unfavorable factors such as noise, missing data, and instability. That is why features in frequency domains, such as eigenfrequency values and mode shapes are also effective damage indicators [2]. For example, a reduction in the fundamental frequency signifies a decrease in the global stiffness of the structures, implying that one or multiple structural members may be compromised as the

global stiffness is obtained by aggregating the individual stiffnesses. Recently, Dang and Nguyen [3] proved that using multiple features from different domains, i.e., statistics, time, and frequency, considerably improves the SHM model performance. In addition, the authors also found that combining multi-modal features with spatial correlation effectively enhances the SHM framework robustness against unfavorable factors [4]. On the other hand, one can directly compare collected vibration signals with reference signals using measurement metrics such as Mahalanobis, L_1 , L_2 , etc. A remarkably large difference implies that the structure may not behave correctly as expected, indicating the occurrence of damage.

With the rapid development of artificial intelligence and data science, numerous structural engineers and scientists have integrated machine learning algorithms with expert knowledge to tackle different structural analysis tasks. Lin et al. [5] extensively compared seven ensemble machine learning models based on various evaluation indicators to perform nonlinear stability assessments of slopes. The authors demonstrated that ensemble models could achieve prediction results with an accuracy of up to 84.74% on a database of 444 slopes, demonstrating that ensemble learning is a promising framework for geoenvironmental problems. Guo et al. [6] developed a novel physical-informed deep learning-based method, namely the deep collocation method, for bending analysis of a wide range of Kirchhoff plates with different geometrical shapes, various boundary conditions and subjected to diverse loads. The results obtained by the proposed method were highly in agreement with exact solutions by analytical or finite element methods. Later, Zhuang et al. [7] explored a deep autoencoder model with the minimum total potential energy principle to analyze the complex behavior of Kirchhoff plates, such as vibration and buckling. Samaniego et al. [8] followed an intriguing research direction of incorporating the energy approach with deep learning to solve the partial differential equations in general computational mechanics. More specifically, the loss functions to train the deep learning models were constructed based on the energy of the mechanical systems. The applicability of the proposed framework was clearly demonstrated via different examples involving linear elasticity, elastodynamics, hyperelasticity, fracture, piezoelectricity, cantilever beam, and plate bending. To perform an analysis of a heterogeneous porous medium effectively and efficiently, Guo et al. [9] developed a stochastic deep collocation method that relied on three components, i.e., physics-informed neural network, neural architecture search, and collocation method. The authors demonstrated that the proposed method consistently outperformed the traditional finite difference method in terms of both accuracy and efficiency.

Besides using these expressible/interpretable indices,

black-box deep learning models are also leveraged to construct precise SHM applications. These models offer the advantage of bypassing the need for preprocessing signal data for extracting features, and raw vibration data can directly enter the SHM model. Next, deep learning model will discard irrelevant factors and learn new representations of data that underlie each operational state. Deep learning model also provides numerous new capabilities such as imputing missing data, reconstructing data from contaminated data, and handling multiple vibration signals simultaneously, etc. However, a major drawback of deep-learning models lies in the requirement of a large volume of labeled data, which is a difficult-to-solve obstacle in real-world scenarios where almost all data are from undamaged conditions, only a small part of the data are collected from temporarily damaged conditions before repair activities are carried out. Additionally, another limitation of deep learning is the time-consuming training process with a number of hyperparameters to be tuned in order to achieve the highest possible accuracy. In summary, in the authors' opinion, these reviewed machine learning/deep learning (ML/DL)-based methods for SHM lack an explicit and interactive mechanism to incorporate expert knowledge to improve their performance during usage.

Large language models (LLMs) are typically large artificial intelligence (AI) models with a large number of parameters, up to the billions, trained on large, general data sets during an extensive training period on massive computational infrastructure. However, when applied to problem-specific tasks, LLM may provide outputs that are counter-intuitive to human understanding. In addition, its answers are notably sensitive to user inputs [10]. Hence, prompt engineering is an emerging methodology that aims at guiding LLMs to provide outputs tailored to specific purposes, concrete requirements, and domain knowledge. However, designing a well-crafted prompt is a challenging task, as a minor modification in wording could have a significant influence on the accuracy and appropriateness of obtained results [11]. For example, utilizing incorrect keywords without a proper understanding of the investigated problems could culminate in non-meaningful results or model failures. As demonstrated in Ref. [12], it is necessary to include domain-specific information in prompts to achieve satisfactory outputs relevant to problems under investigation rather than utilizing general definitions. Hence, prompts should be constructed with a great level of detail, specifying contextual information, objectives, and relevant examples in order to receive useful responses from LLMs [13]. In case of limited data conditions, it is necessary to consider alternative methods such as providing more detailed explanations, including users' experiences [14], or meticulously tuning the wording of prompts to improve response quality [15].

Up to now, prompt engineering has been increasingly integrated in various domains, offering unprecedented benefits. For data extraction, Polak and Morgan [16] developed a framework named ChatExtract, applicable for users without data analysis background. ChatExtract is based on advanced conversational LLMs and prompt engineering to distill informative data with a precision level of up to 90%. For the financial industry, Lo et al. [17] summarized a range of applications using natural language processing (NLP) algorithms and prompt engineering, including areas such as fraud detection, risk management, investment decision-making, and others. The authors also pointed out that although being trained on extensive data, LLMs do not have a comprehensive understanding or common-sense knowledge when working within a specific domain, sometimes resulting in nonsensical answers or providing non-existent facts. In the field of image generation, prompt engineering has led to a significant surge in research on visual prompt engineering. More specifically, the number of publications in the first half of 2023 has tripled compared to 2022. Furthermore, prompt engineering has now become a standard practice for generating images [18] that meet specific criteria including preferred style, selected colors, particular composition, predefined subject, etc. In the realm of chemistry, Hatakeyama-Sato et al. [19] demonstrated that LLMs are capable of providing informative high-level definitions and a general understanding of knowledge; however, they fall short to address specialized contents; thus, it is necessary to resort to other specialized tools or knowledge-infused prompts. For education, the combination of LLMs and prompt engineering offer new opportunities for teaching and learning. For example, teachers can devote more time focusing on the main teaching content while still preparing engaging, interactive slides by leveraging these tools. Meanwhile, students can prompt LLMs to explore better the lessons, such as summarizing the books, raising questions from various perspectives and more [20]. For environmental researchers, who have a large amount of data collected from different sources to analyze, data science skills are indispensable; however, not all environmental researchers who typically work in natural environments, have the time and resources to learn advanced statistics or programming. Thus, LLMs and prompt engineering are great tools for explaining complex mathematical concepts, or unfamiliar programming syntax. Moreover, these tools can even provide ready-to-use code snippets suitable for their specific requirements [21]. On the other hand, for industrial applications, prompt engineering has been a valuable partner for programmers via various new code generation tools such as Copilot, Code Whisperer, etc. These tools excel not only in code generation and completing programming tasks but also in providing

relevant explanations [22]. In fast-paced domains like marketing, in Ref. [23], the authors enumerate numerous advantages brought by prompt engineering, such as facilitating idea exchange, expediting the creation of marketing content, cost reduction, increasing productivity and job satisfaction. In summary, prompt engineering is applicable to multidisciplinary problems, instead of carrying out the time and effort-expensive fine-tuning or retraining process of LLMs for new tasks, we employ well-designed prompts with embedded knowledge to achieve expected results as elaborated in Ref. [24]. Hence, prompt engineering will be an indispensable skill for both AI/non-AI-experts.

In the rest of the paper, Section 2 describes the overall workflow and background knowledge. In Section 3, the proposed framework is applied to different case studies. Finally, Section 4 presents conclusions and proposes some perspectives for the next research stages.

2 Structural health monitoring framework based on large language model and prompt engineering

2.1 Feature extraction from vibration signals

The vibration responses, such as acceleration or displacement of structures subjected to time-varying excitations, are recorded using accelerometers or linear variable differential transformer sensors. With a total number of sensors N_s , a sampling frequency f_s , and a time period of T , we will collect vibration data with the size of $N_s \times L_s$ with $L_s = T \times f_s$. If we monitor the structures over the long-term, there will be N_{measure} different measurements, leading to a database stored in a three dimensional (3D) tensor X with a shape of $N_{\text{measure}} \times N_s \times L_s$.

Based on the governed dynamic equation of the structures presented in Ref. [25], one can derive the relationship between vibration responses and external excitation $F(t)$, as well as, the structural properties like stiffness and mass matrices, as follows:

$$X(t) = \ddot{U}(t) = M^{-1}H \times F(t) - M^{-1}C\dot{U}(t) - M^{-1}KU(t), \quad (1)$$

where $X(t) \in R^{N_{\text{dof}}}$ is the vector of structures' accelerations, N_{dof} is the number of degrees of freedom of the structure, M , C , and $K \in R^{N_{\text{dof}} \times N_{\text{dof}}}$ are the mass, damping, and stiffness matrices, respectively. $F(t) \in R^{N_{\text{dof}}}$ is the vector of applied loads, and H is the input matrix denoting the location of applied loads. It is noteworthy that in practice, due to resource constraints, the number of sensors is usually smaller than the total number of degrees of freedom of the structures $N_s < N_{\text{dof}}$, hence measured vibration data are only a subset of the structures' dynamic responses. Therefore, the location of

sensors is also an important factor to effectively monitoring the operational status of critical structural components. This equation aims to demonstrate the theoretical background of the vibration-based SHM, showing that vibration responses are closely related to the mass and stiffness matrices; hence, if any damage negatively affects the structures' stiffness, it will cause changes in vibration signals. By evaluating these patterns of change and their amplitudes, vibration-based SHM can inversely identify the existence of damages, quantify the damage extent, and localize the damage location. In this study, the following assumptions hold. 1) We consider damages that affect the mass or stiffness matrices of the structure. In other words, vibration-based SHM is not applied for damage types that do not lead to changes in the stiffness matrix of load-bearing structures, for instance, small cracks, surface damage, and defects in secondary components. 2) Vibration signals are adequately collected by suitable vibration sensors and experienced engineers.

However, working directly with multi-dimension vibration signals is impractical, especially when dealing with long signals, i.e., when L_s is great. Note that for current LLMs, each prompt is limited to fewer than 4096 tokens, which corresponds to only four 1024-length signals or just one 4028-length signal. Meanwhile, in practice, there may be a significantly greater number of lengthy signals. That is why, vibration feature extraction is carried out to transform high-dimensional multivariate signals into lower-dimensional feature vectors that underline different structural operations states. The most common features are statistical characteristics such as average, maximum, minimum, standard deviation, and root mean square values. In addition, to describe the distribution form of signal amplitudes, useful statistical features are skewness and kurtosis. Furthermore, the range of amplitude values can be depicted via quartile values, i.e., the median, first, and third quartiles. The third type of feature is spectral features that reflect the signal characteristics in the frequency domain. To obtain these features, we apply a Fourier transformation to the signal and then select the frequency values corresponding to the peaks in the Fourier spectrum. Herein, we focus on the fundamental eigenfrequency value unless otherwise stated.

Let's denote $N_{feature}$ as the total number of extracted features. The feature extraction will convert the database $X \in R^{N_{measure} \times N_s \times L_s}$ into a 3D feature sensor $Z \in R^{N_{measure} \times N_s \times N_{features}}$ with $N_{feature} \ll L_s$.

2.2 Prompt engineering introduction

Prompt engineering is a novel discipline that carefully designs the natural language inputs provided to AI models, such that AI models can effectively deliver

relevant and accurate results. For this purpose, the input prompt should contain contextual information and informative details, that are specifically tailored for desired outcomes (Fig. 1). Note that, this discipline only recently emerged with the development of LLM models that can understand human language and can autonomously interact with humans. With the rapid development of technology and the prevalence of smart devices featuring embedded AI assistants such as Siri and Alexa, prompt engineering represents an exciting frontier for turning AI models into cost-effective, always-available, and information-rich collaborators. However, it is noteworthy that it cannot replace the expertise of experienced engineers or the future judgment of mindful scientists.

Elements of a prompt. A basic prompt typically consists of the following elements (Fig. 2).

1) Context. The context provides the background information, helping LLM understand the problem domain, and follow a particular style and tone. Context can be some keywords, a complete sentence, or an extended paragraph.

2) Instruction/Question. The instruction is a directive given to LLM clearly defining a task to accomplish or a direction for LLM to follow.

3) Constraints. If possible, it should explicitly outline constraints for LLMs in the prompts to narrow the admissible range of responses, or simply to reformulate answers in a user-friendly and readable forms.

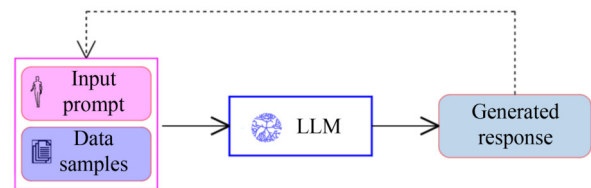


Fig. 1 Graphical representation of prompt engineering with LLM.

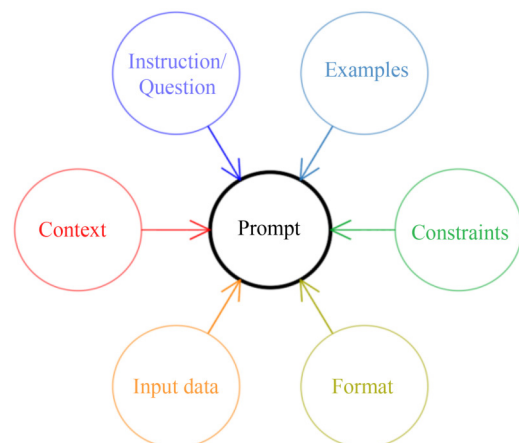


Fig. 2 Typical elements of a prompt.

4) Examples. It is recommended to provide task-specific examples for LLMs to guide LLMs effectively rather than solely relying on too general pre-trained knowledge.

Actually, not all of the above-mentioned elements are requisite for every prompt, and for some problems, other types of elements may be included. An example of a prompt for assessing the state of a structure is shown as follows:

Example of a prompt. An example of a structured prompt is depicted in Fig. 3, in which triple quotation marks are used to denote the beginning and the end of a prompt. The prompt elements, i.e., context, instruction, example, format, and question, are clearly defined. The context element informs LLM to assume the role of an experienced structural engineer assessing the structure’s state based on measured values. Furthermore, structural knowledge is provided in the Instruction element to guide

the LLM in following the criterion of deflection smaller than $1/250$ of the beam length. Input variables for queries are put within curly brackets and denoted by variables such as L_{value}, d_{value} . Some examples are also supplied to quantitatively demonstrate the instruction.

Prompt engineering strategies. Because the responses of LLMs are highly sensitive to input prompts, it should prepare prompts in a strategic way, including distinguishing elements, relevant details, and concrete instructions (Fig. 4).

Role Prompting. This technique assigns LLMs as experts equipped with domain knowledge. For example, in the context of SHM, we assume that LLM is an experienced structural engineer. Subsequently, LLMs will answer with a tone and style of an engineer and furnish responses with a relatively more depth of information compared to a general role.

Zero shot prompting. It is a technique that leverages

```

"""
Context: Let’s assume that you are an experienced structural engineer who assesses the operational state of the structures based on collected vibration data.

Instruction: According to the Design standard, the deflection of a beam should not exceed 1/250 the beam length.

Example:
1. Length: 10 m, Deflection: 20 mm, State: Undamaged
2. Length: 8 m, Deflection: 50 mm, State: Damaged
3. Length: 9 m, Deflection: 30 mm, State: Undamaged

Format: Present the answer in tabular data.

Question: Evaluate the state of the structures with recorded deflection as follows:
Length: {L_value}, Deflection: {d_value}, State: ?
"""
    
```

Fig. 3 Example of a structured prompt.

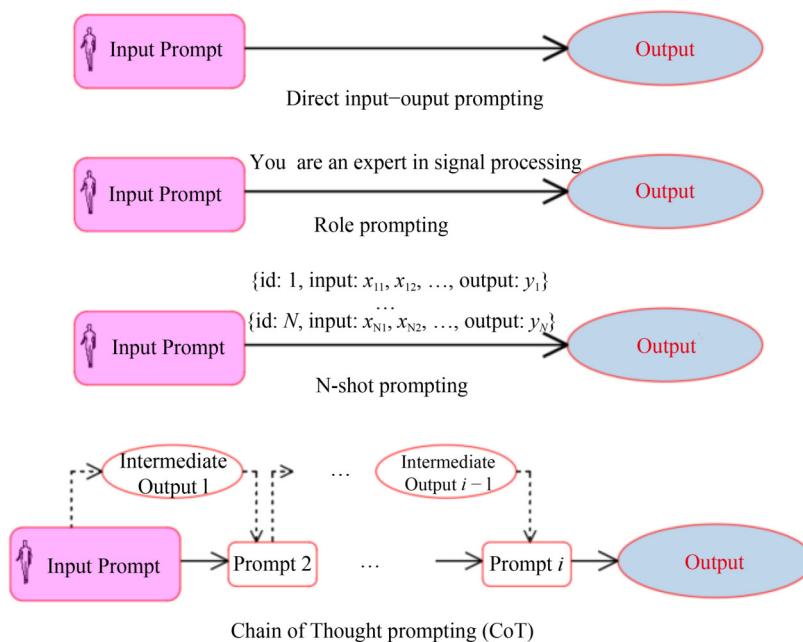


Fig. 4 Schematic representation of different prompt strategies.

LLM's general understanding to immediately address the problem without being given any known example. Since LLM has been trained on extensive database containing up to trillions of tokens. Hence, for some simple and well-known task, it can achieve relatively satisfactory results as LLM may have encountered similar problems during its pre-training process. Moreover, its performance could be enhanced if some meaningful instructions are provided in the prompt.

N-shot prompting. It involves providing LLM multiple examples (*N*) consisting of inputs and corresponding outputs before querying LLM to generate outputs for actual inputs. Note that, even a single example, aka one-shot prompting, can drive LLM provide outputs in a desirable format. *N*-shot prompting is helpful when no instruction can be clearly expressed in natural language to LLM. In other words, the provided examples implicitly serve as conditions/constraints for LLM to follow. An appropriate value for the number *N* of examples varies depending on the problem complexity, *N* can be 3, 5, 10 or more, forming 3-shot, 5-shot, 10-shot prompting, respectively. However, for difficult problems, the required number of examples can be substantial. In some extreme cases, *N*-shot prompting may not be sufficiently effective.

Chain-of-Thought prompting (CoT) [26] decomposes a complex problem into intermediate sub-problems that are solved in a sequential manner providing step-by-step reasoning for final responses. This technique can be expressed in the form of a conversation between users and LLMs. Beginning with an initial prompt, which can be a contextual instruction or a question, LLMs will generate a first response related to the problem of interest. Following this response, users will conduct subsequent instructions/questions to steer LLMs in a wanted direction. The responses of LLMs will progressively improve toward the final purposes while coherently linked to the previous question/answer pairs, forming a logical CoT. Ultimately, the conversation concludes with a final question/response directly addressing the main task of concern. By breaking down the main tasks into intermediate steps and providing additional relevant information, CoT could achieve significantly higher accuracy with even fewer examples compared to other prompting techniques. The CoT technique typically contains the phrase 'Let's think step by step' [26] as a starting point for the thought flow.

2.3 Large language model implementation

Language modeling (LM) refers to a method that predicts subsequent words based on a given context, which has been an active and challenging research topic. LM can potentially enable machines to understand human language. The initial LM models are statistical language

models that are based on Markov assumption and statistical learning methods. However, these methods are unable to handle high-order languages, where the number of probable transitions grows exponentially. With the rise of the neural networks, neural language models based on recurrent neural networks and their variants have been developed. These models are able to address various NLP tasks with long-term dependencies, thanks to their ability to learn relevant features for words and sentences, which is known as representation learning. Recently, the emergence of extensive language databases, aka corpora, and the invention of the Transformer architecture along with its core self-attention mechanism [27] have led to the development of pre-trained language models. These models, initially trained on large-scale corpora before being fine-tuned on downstream tasks, have achieved impressive results on various NLP tasks. The scientists have found that by scaling the size of LMs to incorporate hundreds of billions of parameters, expanding the volume of corpora up to hundreds of gigabyte (Gb) or even several terabytes, and conducting training on thousands of the latest GPUs, it is possible to elevate LM performance to a new level, even achieving superhuman capabilities. These LMs are referred to as LLMs. Interestingly, LLMs feature the so-called emerging ability which refers to the fact that LLMs can achieve surprisingly good results on entirely new tasks based on only a few given examples (few-shot learning) or even without any trained data (zero-shot learning). Moreover, LLMs can perform effectively on non-language tasks such as numerical computation or programming.

Examples of popular LLMs are Google FLAN-T5 which was released in 2022 [28]. Its sizes range from 60M (FLAN-T5 small) to 11B (FLAN-T5 XXL). This model is recognized for its balance between speed, efficiency, and performance; thus it is preferred by some authors for real-world application. Besides, the most popular LLM is the Generative Pre-training Transformer model from OpenAI [29]. This model is constructed based on a very deep causal-transformer deep learning architecture, for instance, GPT3.0 has a size of 175 billion parameters. The GPT models have achieved numerous state-of-the-art results in various domains including mathematics, physics, and chemistry in addition to NLP tasks and surpassed all LMs at that time. It is noteworthy that GPTs can take very long inputs, for example, GPT-3, GPT-3.5, and GPT-4 can handle inputs with maximum lengths of 2049, 4096, and 32000 tokens, respectively. In terms of outputs, they can generate responses with a length of several hundred pages. It is recalled that GPTs are commercial and closed-source models. On the other hand, recently, Meta has released an open-source and free LLM, namely, LLAMA2 [30] which is able to produce competing performances with closed LLMs. Since LLAMA2 is an open-access model;

thus, it is very flexible to customize and integrate LLAMA2 into both online and offline applications, on a low-budget, unlike closed LLMs which only operate online via paid application programming interface (API).

To deploy LLMs, there are two major ways: public APIs and directly running LLMs on a local machine. The first way is employed by commercial closed-source LLMs, such as GPT, for which users do not need a decent computer, they simply need to subscribe to an account, receive API keys, and then connect to the web servers of LLM providers. Each user’s query will be handled by computation servers, and the answers will be sent back to the users. This method is fairly fast thanks to the modern computation servers, but each query is charged by the LLM providers, making it quite expensive as the number of queries quickly increases. The second way to run LLMs is to download open-source LLMs and pre-trained checkpoints on personal computers, and then run them as normal functions. However, LLMs are usually large in size; for instance, the size of the 7 billion parameter-version of Llama 2 is 13.5 GB. Thus, a specialized technique, called quantization [31], is resorted to reduce the size of local LLMs and shorten the inference time while still maintaining the result accuracy to some extent. The essential idea of this method is to lower the precision of trained models’ parameters to 3-bit, 4-bit, or 8-bit instead of the original 16-bit. By doing so, one can reduce the model size, and the demand for dynamic memory, and expedite the inference process.

On the other hand, current LLMs still suffer from some limitations such as the demand for massive computation resources and effort for training. For example, training GPT-3 cost several million dollars, while training LLAMA2 required hundreds of thousands of GPU hours, and consumed several hundred watts of power. Another

limitation is that LLMs may yield toxic, biased, untrue responses or outdated information because they are unable to autonomously update their knowledge. Therefore, it is necessary to mitigate the above problem and guide LLMs toward correct answers by incorporating human knowledge and/or combining LLMs with other external programs.

2.4 Workflow of the proposed structural health monitoring framework

A high-level workflow of the proposed LLM-assistant SHM framework is schematically depicted in Fig. 5 which can be roughly described in four stages. The first stage is standard data management, similar to other SHM frameworks, where collected vibration data are indexed, labeled with structural states, and stored in a structural database. In this study, each data sample is stored in the JSON format, consisting of multiple attribute-value pairs: the first pair for the data indices, the second pair for structural state, and the third pair for sensor signals, such a format can be straightforwardly converted into a dictionary in Python. The entire vibration data set can be expressed as a list of dictionaries, with the length of the list corresponding to the total number of samples. The second stage is to implement LLMs using one of the two methods mentioned above, i.e., via an API service or a local model with quantization. In terms of computational devices, the authors utilize the ecosystem provided by Google: data storage with Google Drive, calculation with Google Colab thanks to its data security, practicability, good performance with the GPU A100, and 32 GB of RAM for LLM inference. In the third stage, one poses various prompts to explore the structural databases, perform feature extraction, and construct a classifier.

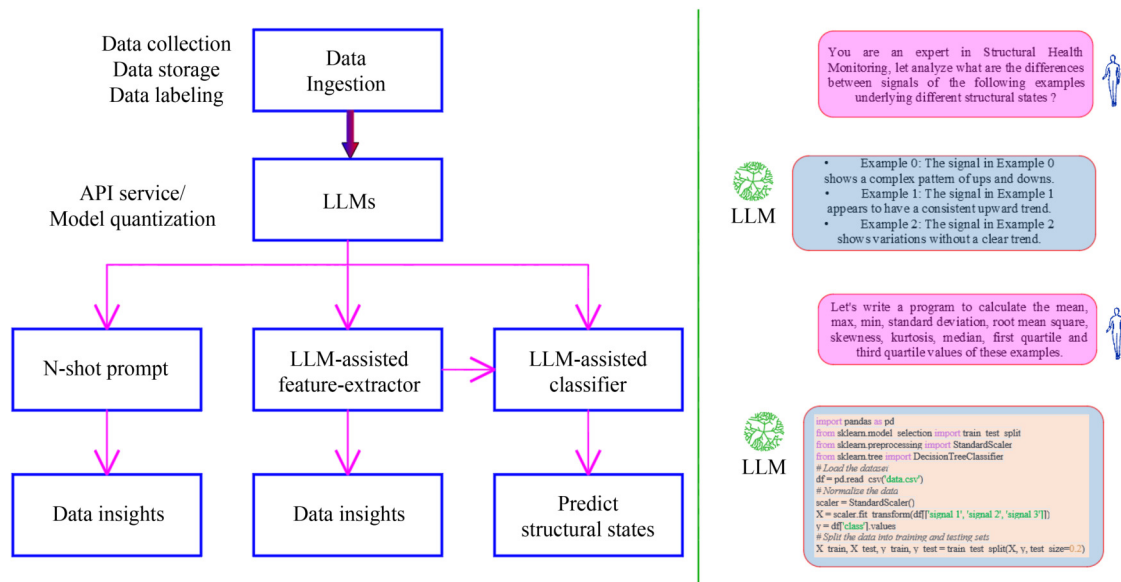


Fig. 5 Working flow of the SHM framework with LLM and prompt engineering.

When receiving prompts, LLMs generate responses consistently related to given data and users' questions. One can provide some examples of different vibration signals and their associated structural states, to assist LLMs in understanding better about the investigated problem. LLMs could yield some insights about vibration data characteristics that may underlie structural states, helping distinguish them together. On the other hand, LLMs are acknowledged for their programming assistance ability because they are pre-trained on a large code corpus; hence, one prompts LLMs to build a feature extractor for vibration signals that LLM have seen via N-shot prompts. With the extracted features, one can prompt LLM to unveil which features signify damaged/abnormal states. One can also prompt LLMs to build a ML-based classifier to predict the actual states of structures.

In terms of prompts for the SHM application, we employ the CoT prompt strategy, which typically consists of the following prompts.

1) The first prompt involves the role prompting technique, assuming LLM is a structural engineering/signal processing expert to analyze the vibration data. By doing so, LLM will produce outputs containing keywords and contents relevant to structural dynamic fields such as frequency, magnitude, oscillation, and noise.

2) Next, in the second prompt, we define the key characteristics that help assess the structures' states according to the engineers' preferences. After that, we prompt LLM to write code snippets to extract desirable characteristics automatically. These prompts incorporate the SHM knowledge of users into the framework, converting long vibration signals to low-dimension vectors of features. It is noted that deciding what features to extract is a critical step that has a profound impact on damage detection performance. More details on the feature extraction for SHM can be found in Ref. [32].

3) Afterwards, the LLM is prompted to provide qualitative assessments of the underlying characteristics of feature vectors for every structural state. Since the size of input data has been significantly reduced, thus we can include some input examples within the maximum length of a prompt and then ask the LLM to provide overview assessments on informative characteristics of feature vectors that may underlie specific structures' states.

4) For the fifth prompt, we query LLM to produce code snippets to build a machine-learning model to classify feature vectors in corresponding structures' states. Based on the previous prompts, LLM has acquired an understanding of various structures' states, data format, and feature vectors; however, to generate a workable ML-based classifier, we need to clarify every step of the development of a machine learning model, including importing necessary libraries, loading data, standardization, data splitting, model selection, training process.

5) Subsequently, LLM is prompted to produce code

snippets to evaluate the classifier performance. Herein, we can specify which evaluation metrics are preferred, like f1-score, accuracy, or receiver operating characteristic (ROC) curves, and prompt LLM to generate code snippets for evaluating the ML-based classifier on the testing data set created in the previous steps.

It is noteworthy that, the present SHM workflow minimizes the need for manual statistic calculation and programming from scratch; thus, relieving related burdens, such as debugging and code testing. This helps researchers and engineers focus more on performing data analysis and analyzing the behaviors of structures.

3 Application examples

3.1 American Society of Civil Engineers (ASCE) benchmark steel frame structure

The first application example is the renowned benchmark steel frame structure with braces at the University of British Columbia [31], as illustrated in Fig. 6. This publicly accessible data set has been utilized by various authors [32,33] to validate their SHM methods. It is utilized here to demonstrate the applicability of the proposed method to facilitate reproducibility. The structure was subjected to white noise random excitation; its vibrations were measured through sensors installed across all stories. For this first example, we only utilize a single sensor on the first floor to demonstrate the working mechanism of the proposed method. The collected vibration signals are divided into shorter time series and stored in a two dimensional tensor with a shape of [1735, 100] with 1735 being the total number of samples, and 100 being the signal length. The outputs of interest are labeled with 9 structural states which correspond to the following: intact/undamaged state (State 0), removal of the east side braces (State 1), removal of the braces on all floors on a bay in the south-east corner (State 2), removal of braces on the 1st and 4th floors on a bay in the south-east corner (State 3), removal of braces on the 1st floor on a bay in the south-east corner (State 4), removal of braces

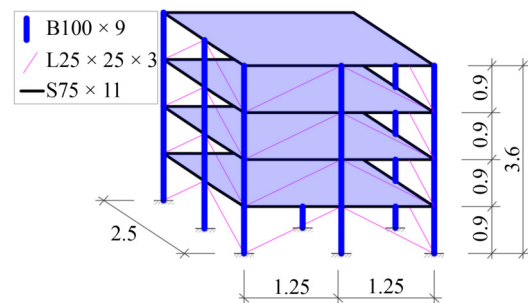


Fig. 6 Graphical representation of the ASCE benchmark steel frame structure.

on all floors on the east face and on the 2nd floor on the north face (State 5), removal of all braces on all faces (State 6) removal of all braces and loosening bolts on all floors of beams on the east face (State 7), removal of all braces and loosening bolts on the first and second floors of beams on the east face (State 8). More details about the experiment can be found in Ref. [34].

Next, we provide LLM with 9 examples, one for each class, and demand LLM to automatically perform data analytics under the assumption that LLM is an expert in signal processing, according to the role prompting strategy. These examples are graphically illustrated in Fig. 7 and the responses of LLM are shown in Fig. 8. It can be seen that LLM provides about three assessments for each signal, mainly focusing on frequency oscillation and signal amplitude, aligning with signal patterns (Fig. 7). These insights are relatively general, hence, not very useful for further damage detection. Via these examples, LLM becomes acquainted with the problem, has memory of the data format, and is assigned an expert role.

After that, we instruct LLM to perform data extraction, converting high-dimensional vibration data into low-dimension feature vectors. In the authors' opinion, this step can be conducted in two ways. The first way is to directly demand LLM to perform mathematical operations to obtain statistical features. Despite LLM's ability to perform mathematical operations with relatively good accuracy, each prompt takes seconds to accomplish.

Thus, when working with thousands of examples, the total computation time could be several hours or days. Furthermore, the authors experienced a reduction in accuracy as the number of samples increased. The second way involves instructing LLM to act as a programmer and generate a program for feature extraction, given that the desired features are clearly mentioned in the prompt. Surprisingly, the code generation by LLM, depicted in Fig. 9, smoothly operates without any bugs. We also require LLM to format the outputs in a list of dictionaries and save them in a JSON file with ID, state, and feature fields, following a data management technique used for raw vibration data.

Subsequently, LLM is utilized to analyze the underlying patterns of feature vectors that may lead to different structural states. Some feature vectors with assigned classes are fed to LLMs, the meaning of features is also clarified, and LLM is considered as an expert data analyst adept at classifying data. The prompt and LLM's response is demonstrated in Fig. 10. Apparently, LLM examines classes one by one, discussing the properties of the value ranges and value distributions. These discussions serve as the basis for qualitatively discriminating data samples within various groups.

Since, LLM has explored the structural database and acquired an understanding of the structural problem to some extent via multiple rounds of prompts and also gains knowledge about data management, including input features and desired outputs, we prompt LLM to build a

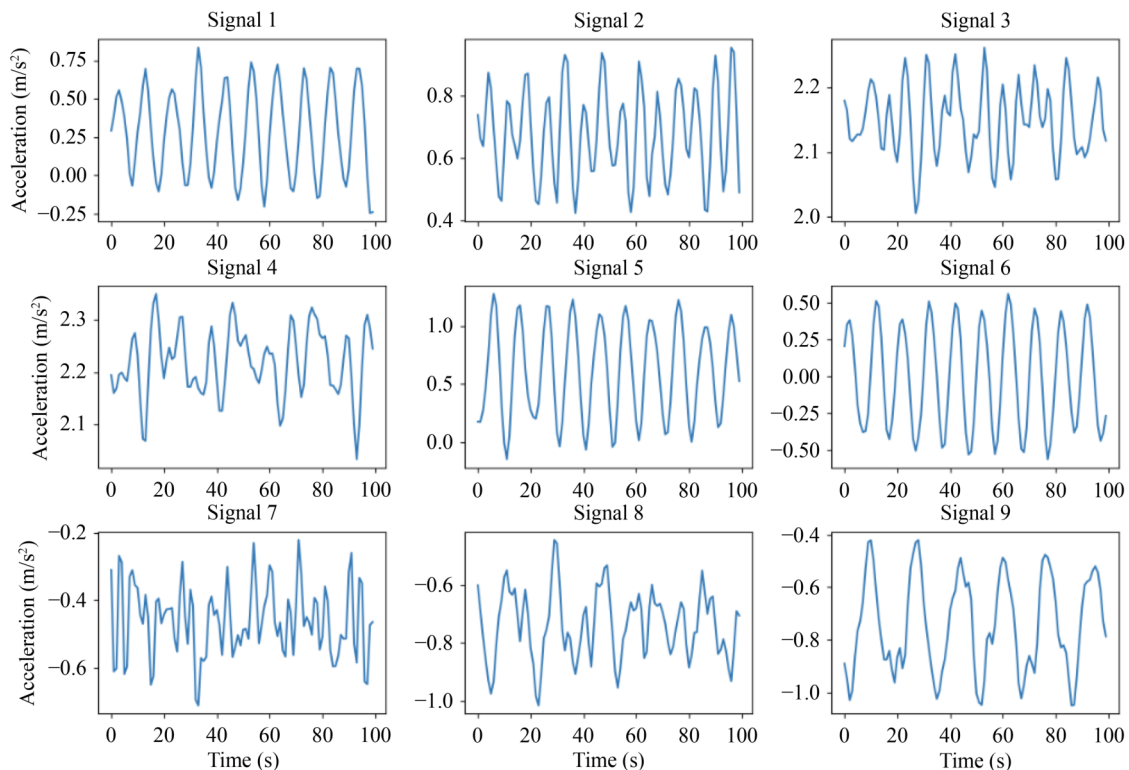


Fig. 7 Example of vibration signals corresponding to nine structural states.



Fig. 8 Prompt and LLM's corresponding responses for analyzing raw vibration signals using the GPT 3.5 model.

machine learning-based classifier to identify the structural states, assuming that LLM possesses expertise in both SHM and machine learning. The prompts and LLM's corresponding responses are demonstrated in Figs. 11 and 12. The program provided by LLM operates straightforwardly with the investigated database without

requiring any additional steps for manual correction or adjustment. Notably, this classifier achieves an average high accuracy of 98% across all classes (Table 1).

In summary, this case study clarifies all the steps of the workflow presented in Fig. 5, demonstrating the applicability of the LLM-assisted SHM framework with

PROMPT INPUT

Instruction: Write a program to calculate the mean, max, min, standard deviation, root mean square, skewness, kurtosis, median, first quartile and third quartile values of the signals of these examples.
Format: Present the results in a table with rows representing the examples, the first column denoting the example id, the second column indicating the example label, and other columns displaying calculated values.
Constraint: No need further explanations.

LLM RESPONSE (GPT 3.5)

```

import numpy as np
import pandas as pd
from scipy import stats

# Define a function to calculate the requested statistics for a given signal
def calculate_statistics(signal):
    mean = np.mean(signal)
    maximum = np.max(signal)
    minimum = np.min(signal)
    std_deviation = np.std(signal)
    rms = np.sqrt(np.mean(np.square(signal)))
    skew = stats.skew(signal)
    kurt = stats.kurtosis(signal)
    median = np.median(signal)
    first_quartile = np.percentile(signal, 25)
    third_quartile = np.percentile(signal, 75)
    return [mean, maximum, minimum, std_deviation, rms, skew, kurt, median,
            first_quartile, third_quartile]

# Create an empty DataFrame to store the results
results_df = pd.DataFrame(columns=["Example ID", "Example Label", "Mean", "Maximum", "Minimum", "Standard Deviation", "RMS", "Skewness", "Kurtosis", "Median", "First Quartile", "Third Quartile"])

# Iterate through each example
for example in examples:
    example_id = example["id"]
    example_label = example["class"]
    signal_values = example["signal"]

    # Calculate statistics for the signal
    statistics = calculate_statistics(signal_values)

    # Append the results to the DataFrame
    results_df = results_df.append({"Example ID": example_id, "Example Label": example_label, "Mean": statistics[0], "Maximum": statistics[1], "Minimum": statistics[2], "Standard Deviation": statistics[3], "RMS": statistics[4], "Skewness": statistics[5], "Kurtosis": statistics[6], "Median": statistics[7], "First Quartile": statistics[8], "Third Quartile": statistics[9]})

```

Fig. 9 Prompt and LLM's corresponding responses for feature extractions using the GPT 3.5 model.

detailed results, showing that this framework relieves the users from the burden of conducting data exploration by hand or building a task-specific program from scratch.

3.2 Laboratory 3-story steel frame

The second case study involves a laboratory three-story steel frame at the Alamos laboratory [35], as shown in Fig. 13. The structure was subjected to random excitation caused by a shaker positioned at its base, and vibrations were recorded at the three stories plus the base through four accelerometers. For this example, the sliding window technique was utilized to divide the measured signals into sub-time-series. These sub-time-series were subsequently reshaped into a 3D tensor with a shape of

[35328,4,128] where 35328 is the total number of data samples, 4 is the number of sensors, and 128 is the signal length. The data samples were labeled by one of seven structural states as detailed in Table 2. Damage is simulated by reducing the stiffness of the frame's columns. In this example, we consider seven structural states, which are an undamaged state (State 0), 50% stiffness reduction of one column on the first floor (State 1), 50% stiffness reduction of two columns on the first floor (State 2), 50% stiffness reduction of one column on the third floor (State 3), 50% stiffness reduction of two columns on the first floor (State 4), 50% stiffness reduction of two columns on the second floor (State 5), 50% stiffness reduction of one column on the second floor (State 6). A more detailed description of the

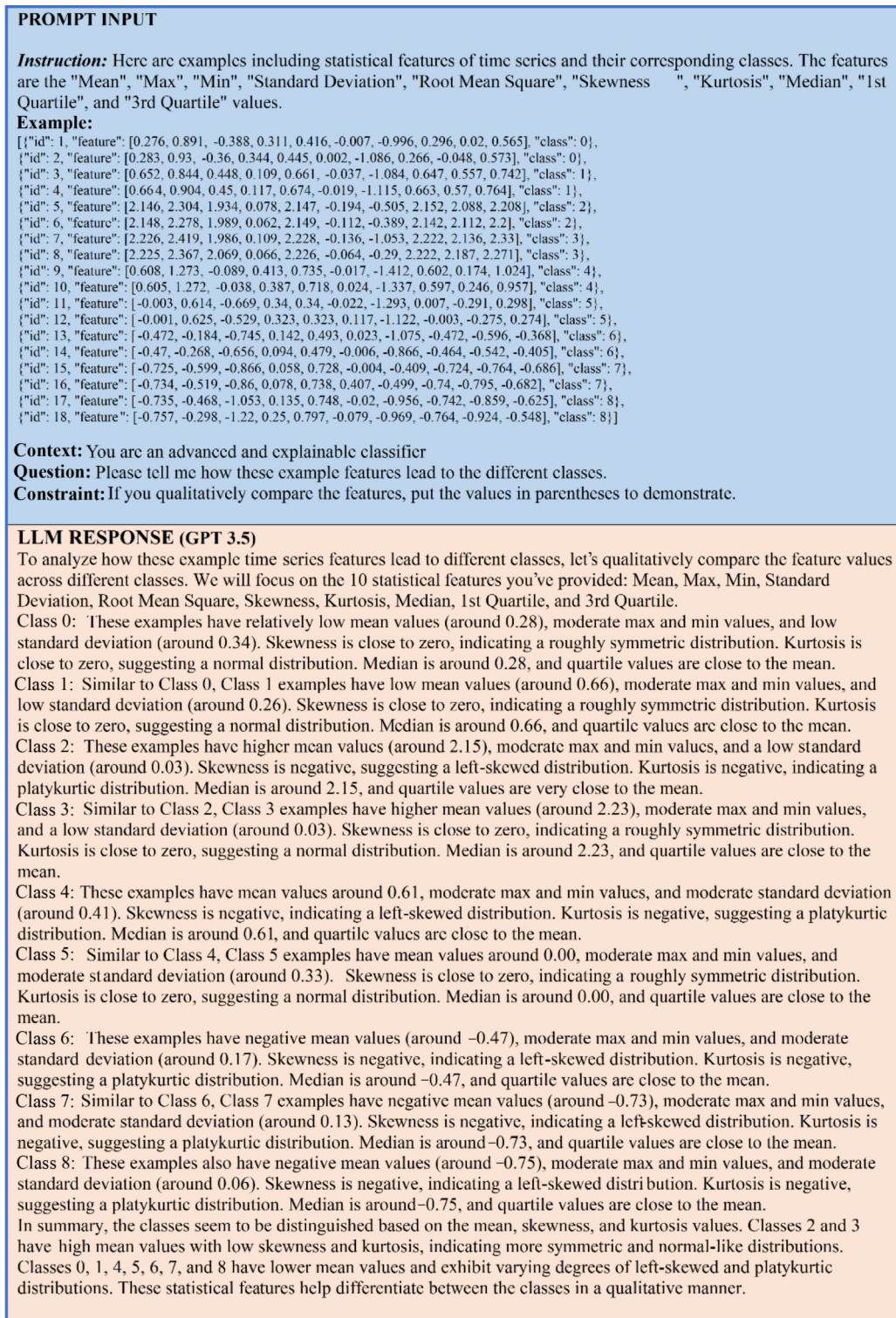


Fig. 10 Prompt and LLM's corresponding responses for classifying structural states using extracted features using the GPT 3.5 model.

experiment can be found in Ref. [35] and vibration data can be publicly accessed on the website of the Alamos laboratory. This is one of the most well-known benchmark data sets for SHM applications involving multivariate vibration signals [36, 37].

Similarly to the first case study, a sequence of prompts

is posed to LLM. On the one hand, the LLM-based framework provides meaningful insights into the dynamic characteristics of vibration signals that may underlie different structural states (Fig. 14). On the other hand, the prompt progressively steers the LLM to understand the data format and the problem under investigation. More

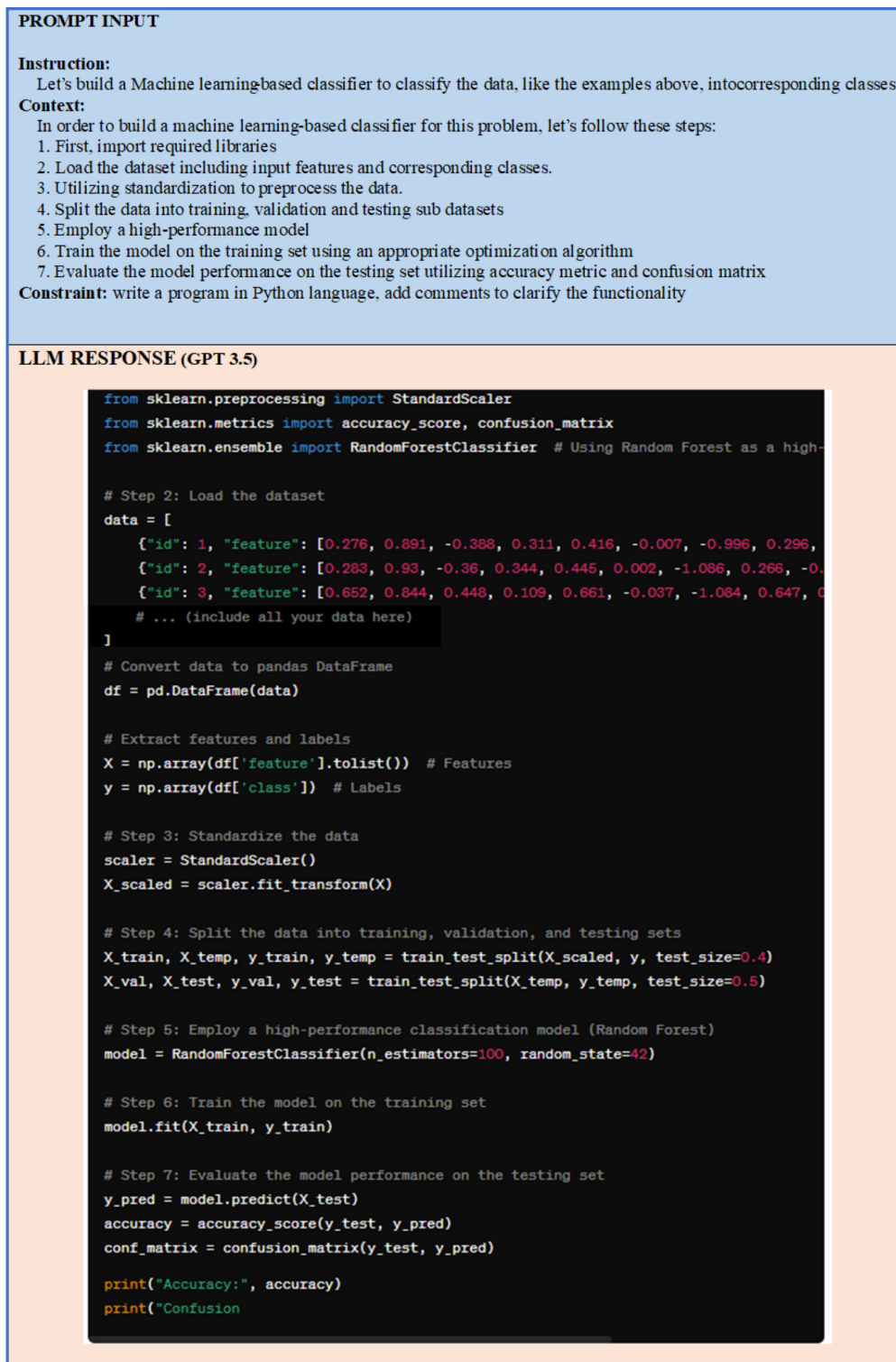


Fig. 11 Prompt and LLM's corresponding responses for building a ML-based classifier using the GPT 3.5 model.

specifically, in the first prompt, we directly provide some data samples including vibration signals from four sensors and their corresponding structural states, i.e., the labels of data samples. We also instruct LLM to extract informative properties that may exist within raw vibration signals. Via these examples, LLM comprehends the data's

nature and format; enabling it to deliver an error-free program for extracting meaningful dynamic features. Afterward, we provide certain examples of extracted data to LLM and demand it to perform feature analysis, unveiling which features and how their values could assist users in discriminating different structural states.

PROMPT INPUT

Instruction:
Let's write a program to fine tune the above machine learning classifier

Context:

1. Define some key hyperparameters such as `n_estimators`, `learning_rate`, `max_depth`, `min_child_weight`,
2. Perform Random Search
3. Re-Train the Model with Best Hyperparameters
4. Re-evaluate the fine-tuned Model

Constraint: write a program in Python language, add comments to clarify the functionality

LLM RESPONSE (GPT 3.5)

```

# Step 1: Define key hyperparameters
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Step 2: Perform Random Search
from sklearn.model_selection import RandomizedSearchCV

rf = RandomForestClassifier(random_state=42)
random_search = RandomizedSearchCV(estimator=rf, param_distributions=param_grid, n_iter=100)
random_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = random_search.best_params_
print("Best Hyperparameters:", best_params)

# Step 3: Re-train the model with Best Hyperparameters
best_rf = RandomForestClassifier(**best_params, random_state=42)
best_rf.fit(X_train, y_train)

# Step 4: Re-evaluate the fine-tuned Model
y_pred = best_rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy after fine-tuning:", accuracy)
print("Confusion Matrix after fine-tuning:")
print(conf_matrix)

```

Fig. 12 Prompt and LLM's corresponding responses for fine-tuning a ML-based classifier for the ASCE benchmark data set.

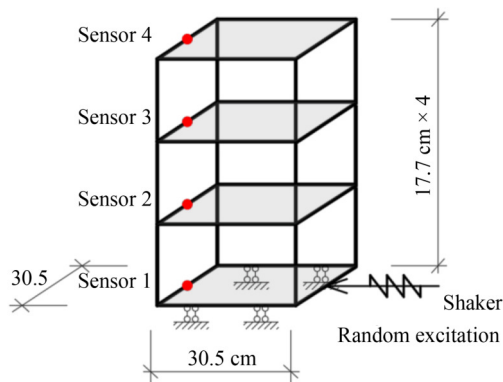
Finally, we instruct the LLM to develop an ML-based classifier for inversely detecting corresponding structural states given vibration signals' feature vectors. In the following paragraphs, the results of these prompts will be presented and discussed.

First, the maximum number of tokens for each prompt

for LLMs is 4096, whereas, each data sample in this example contains 4×128 -length signals and corresponding labels, i.e., only a limited number of data samples can be included in each prompt. It is also noted that each numerical value should be represented with no more than three decimal digits; otherwise, the maximum length of a

Table 1 Number of trainable parameters of the proposed framework

Class	Precision	Recall	F1-score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	1.00	1.00	1.00
6	1.00	1.00	1.00
7	0.95	0.95	0.95
8	0.97	0.97	0.97

**Fig. 13** Schematic representation of the three-story steel frame structure at the Alamos laboratory [35].**Table 2** Investigated structural states of the three-story steel frame

State	Structural state description
0	healthy
1	reducing 50% stiffness of one column on the first floor
2	reducing 50% stiffness of two columns on the first floor
3	reducing 50% stiffness of one column on the third floor
4	reducing 50% stiffness of two columns on the third floor
5	reducing 50% stiffness of two columns on the second floor
6	reducing 50% stiffness of one column on the second floor

prompt would be exceeded. Note that LLMs faced difficulty when directly working with raw vibration signal, as they consider each digit instead of each real value as a token, making the number of token very large. Specifically, when including seven data samples in a single prompt, each representing a structural state, the total number of characters goes beyond 30000. Consequently, such an excessively lengthy prompt cannot be accepted by current LLMs. That is why we investigate sensor signals story by story. The responses of LLM are summarized in Fig. 15. It can be seen that LLM summarized the range of signal amplitudes, and qualitatively assessed the oscillation/variations of the

signals. However, this information remains still vague hence we instruct LLM to apply advanced signal processing techniques and feature extraction for further analysis.

Next, for feature extraction, the code provided by LLMs can be used in a straightforward way without requiring any corrections, as presented in the previous example, except that we include the fundamental frequency and the length of feature vector of each sample is 44 for all four sensor signals. We provide seven extracted features to LLM, each for a structural state, assuming LLM as an advanced classifier, and prompt it to qualitatively analyze these features. Next, for feature extraction, the code provided by LLMs can be used in a straightforward way without requiring any corrections, as presented in the previous example, except that we include the fundamental frequency (Fig. 16). The length of the feature vector of each sample is 44 for all four sensor signals. We provide seven examples of extracted features to LLM, one for each structural state, assuming LLM as an advanced classifier, and prompt it to qualitatively analyze these features.

Although LLMs are powerful foundation models that can understand natural language texts at an impressive level, LLMs themselves are regarded as black boxes due to their complex, very deep architecture with many parameters. Hence, it is difficult for most users to comprehend the internal working mechanism of LLMs, discern the rationale for selecting a specific machine learning model over others, determine the reliability of the outputs, and assess whether there are any bugs/errors within them. One practical and helpful way to promote the interpretability and transparency of LLM-assistant frameworks is to include the steps of reasoning within the prompt. These steps provide LLMs knowledge on different aspects involving data under investigation and machine learning expertise, making the model outputs less opaque and more reliable. Herein, to build a classifier for identifying the structural states of the investigated example, we employed detailed prompts (Figs. 17 and 18) explicitly specifying data loading, data standardization, data splitting, the definition of the ML classifier, the training process, fine-tuning and the evaluation. By utilizing these code snippets, the SHM accuracy for this example reaches around 90%, with more details shown via the confusion matrix and reports presented in Table 3 and Fig. 19. To graphically demonstrate the obtained structural identification results, the One-vs-Rest Receiver Operating Characteristic curve and corresponding Area Under the Curve (AUC) are utilized to demonstrate the obtained structural identification results graphically. The prompt and corresponding snippet code derived from the GPT 3.5 answer is presented in Fig. 20, and resulting ROC curves are illustrated in Fig. 21. It is recalled that a random guess has an AUC of 0.5, whereas a perfect

ROMPT INPUT			
<p>Context: You are an expert in signal processing.</p> <p>Instruction: please tell me what the differences are in the signals of these samples that lead to different classes?</p>			
<p>Signal 1 {"id": 0, "state": 0, "signal 1": [0.0621,-0.0842,-0.1702...]}, {"id": 1, "state": 1, "signal 1": [0.1213, 0.8667, 0.2382...]}, {"id": 2, "state": 2, "signal 1": [0.8327,-0.4824,-0.9297...]}, {"id": 3, "state": 3, "signal 1": [-0.2737,-0.6252,-0.0177...]}, {"id": 4, "state": 4, "signal 1": [-0.8071,-0.0999, 0.5128...]}, {"id": 5, "state": 5, "signal 1": [-0.3478,-0.2582, 0.3599...]}, {"id": 6, "state": 6, "signal 1": [-0.6444,-0.4063,-0.2579...]}</p>	<p>Signal 2 {"id": 0, "state": 0, "signal 2": [0.2879, 0.1407,-0.2131...]}, {"id": 1, "state": 1, "signal 2": [-0.114, -1.3133,-1.0262...]}, {"id": 2, "state": 2, "signal 2": [-0.3578,-0.0309, 0.3185...]}, {"id": 3, "state": 3, "signal 2": [0.8137,-0.1818,-0.901...]}, {"id": 4, "state": 4, "signal 2": [0.3534, 0.1119,-0.1116...]}, {"id": 5, "state": 5, "signal 2": [-0.0227, 0.2575, 0.2918...]}, {"id": 6, "state": 6, "signal 2": [0.0291, 0.0431,-0.0568...]}</p>	<p>Signal 3 {"id": 0, "state": 0, "signal 3": [-0.6065,-0.032, 0.6263...]}, {"id": 1, "state": 1, "signal 3": [-0.4903, 0.0066, 0.1535...]}, {"id": 2, "state": 2, "signal 3": [-0.3543, 0.2196, 0.5267...]}, {"id": 3, "state": 3, "signal 3": [-0.1999, 0.716, 0.7671...]}, {"id": 4, "state": 4, "signal 3": [0.4378, 0.3504,-0.0052...]}, {"id": 5, "state": 5, "signal 3": [0.3578, 0.2242,-0.1262...]}, {"id": 6, "state": 6, "signal 3": [-0.0995, 0.3882, 0.5607...]}</p>	<p>Signal 4 {"id": 0, "state": 0, "signal 4": [0.2915,-0.0103,-0.2594...]}, {"id": 1, "state": 1, "signal 4": [0.4269, 0.4886, 0.3303...]}, {"id": 2, "state": 2, "signal 4": [0.4325,-0.0805,-0.4601...]}, {"id": 3, "state": 3, "signal 4": [-0.0552,-0.4152,-0.2994...]}, {"id": 4, "state": 4, "signal 4": [-0.3237,-0.2781,-0.0003...]}, {"id": 5, "state": 5, "signal 4": [-0.2552,-0.232, 0.0338...]}, {"id": 6, "state": 6, "signal 4": [0.3982, 0.0142,-0.2087...]}</p>
LLM RESPONSE (GPT 3.5)			
<p>Signal 1 State 0 : relatively small fluctuations, no strong periodicity or clear patterns in the signal, values between around -1.23 and 1.47. State 1 : larger variations compared to State 0 , more oscillations and fluctuations, values between -1.04 and 1.15. State 2: even larger variations and oscillations, complex patterns, values between approximately -1.13 and 1.20. State 3 : a mix of small and large fluctuations, some periodicity, values between approximately -1.26 and 1.20. State 4 : significant fluctuations, periodic patterns with alternating peaks and valleys, values between -1.32 and 1.15. State 5 : moderate fluctuations, some periodicity, but less regular compared to State 4 , values between -1.34 and 1.14. State 6: significant variations and oscillations, periodic patterns with alternating peaks and valleys, values between approximately -1.34 and 1.34.</p>	<p>Signal 2 State 0 : oscillations of varying amplitudes and frequencies, with occasional sharp spikes, amplitudes ranging from approximately -1.3133 to 1.0975. State 2: relatively smooth and less noisy compared to State 1, amplitudes ranging from approximately -0.8903 to 0.9158. State 3 : larger fluctuations compared to Classes 17 and 18, with relatively high-frequency components, amplitudes ranging from approximately -1.2772 to 1.609. State 4: relatively smooth and gradual variation, resembling a slowly oscillating wave, amplitudes ranging from approximately -1.9437 to 1.7176. State 5 : periodicity with a repeating pattern, sinusoidal-like behavior with regular oscillations, amplitudes ranging from approximately -0.7391 to 0.9536. State 6 : complex and irregular pattern, more chaotic compared to the other classes, amplitudes ranging from approximately -0.7391 to 0.9536.</p>	<p>Signal 3 State 0 : have some level of noise or random fluctuations, no clear periodicity nor a specific pattern. State 1 : relatively consistent periodic oscillations and smooth compared to State 0 , with fewer sudden changes or noise. State 2 : relatively stable patterns with gradual variations, fewer rapid changes compared to State 0 . State 3 : has a clear and consistent periodicity with a noticeable frequency. State 4: relatively smooth and lacks significant noise or rapid changes. State 5 : more complex pattern with alternating positive and negative peaks. State 6: highly irregular with sharp spikes and rapid transitions, more chaotic compared to the other classes. State 0 lacks a clear pattern and has more noise compared to the others. State 1 has a distinct alternating pattern, and State 2 exhibits a relatively stable and gradual variation. Classes 21 and 22 have consistent periodicity, while State 6 is highly irregular and chaotic.</p>	<p>Signal 4 State 0 : exhibits a relatively complex pattern with alternating peaks and valleys. State 1: The signal values are mostly positive, more regular compared to Sample 0 . State 2: has a pattern of oscillation similar to Sample 1 but with variations. State 3: have a more irregular and complex pattern compared to Samples 1 and 2. State 4: The signal values are close to zero, relatively flat and less oscillatory compared to the previous samples. State 5: The signal values are mostly small and clustered around zero. There is a minor oscillatory pattern but with small amplitude variations. State 6: exhibits a more significant oscillatory pattern with larger amplitude variations. It's important to note that these observations are based solely on the "signal 4" feature, and the actual differences between classes may involve more complex patterns and relationships that are not immediately apparent from the raw data.</p>
<p>In summary, the key differences between the signal classes appear to be the magnitude of fluctuations, the presence or absence of periodicity, and the overall complexity of the patterns. Different classes exhibit varying degrees of signal variations and patterns, which likely contribute to the classification into different classes. Further analysis and feature extraction techniques can be applied to extract more specific differences and patterns for classification purposes.</p>			

Fig. 14 Prompt and LLM’s corresponding responses for analyzing raw vibration signals of the laboratory frame structure using the GPT 3.5 model.

prediction attains an AUC of 1.0. It can be seen that the AUC obtained in this example by the LLM-based framework achieves AUC values of more than 0.97, reaffirming good damage detection results.

On the other aspect, in the authors’ opinion, current AI

tools, even the latest large-scale foundation models, are still not able to directly deal with complex dynamics such as permanent displacement or hysteresis for at least two reasons. 1) First, these AI tools are trained on general databases which contain a wide range of types of data.

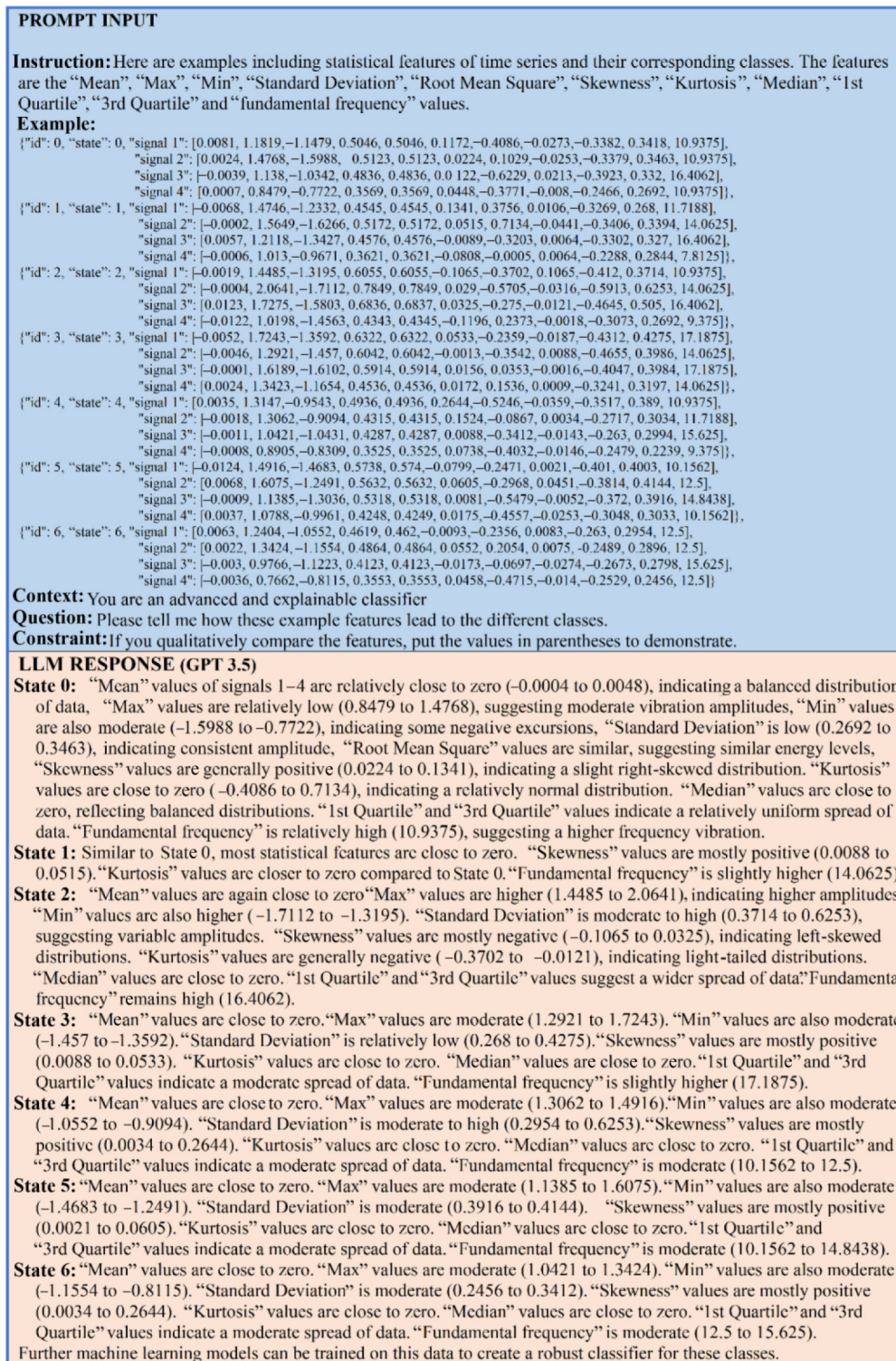


Fig. 15 Prompt and LLM’s corresponding responses for classifying structural states using extracted features of the laboratory frame structure using the GPT 3.5 model.

However, such databases may lack sufficient complex dynamic data, which are difficult and scarce to obtain. Only a small number of research groups possess these kinds of data. 2) Even with these relevant data in hand, fine-tuning large-scale AI foundation models for a civil

engineering task requires expensive computation servers, long training time, and advanced knowledge in deep learning architecture, which is not readily available to most structural engineers. However, LLM-based AI models can be used indirectly, such as assisting in

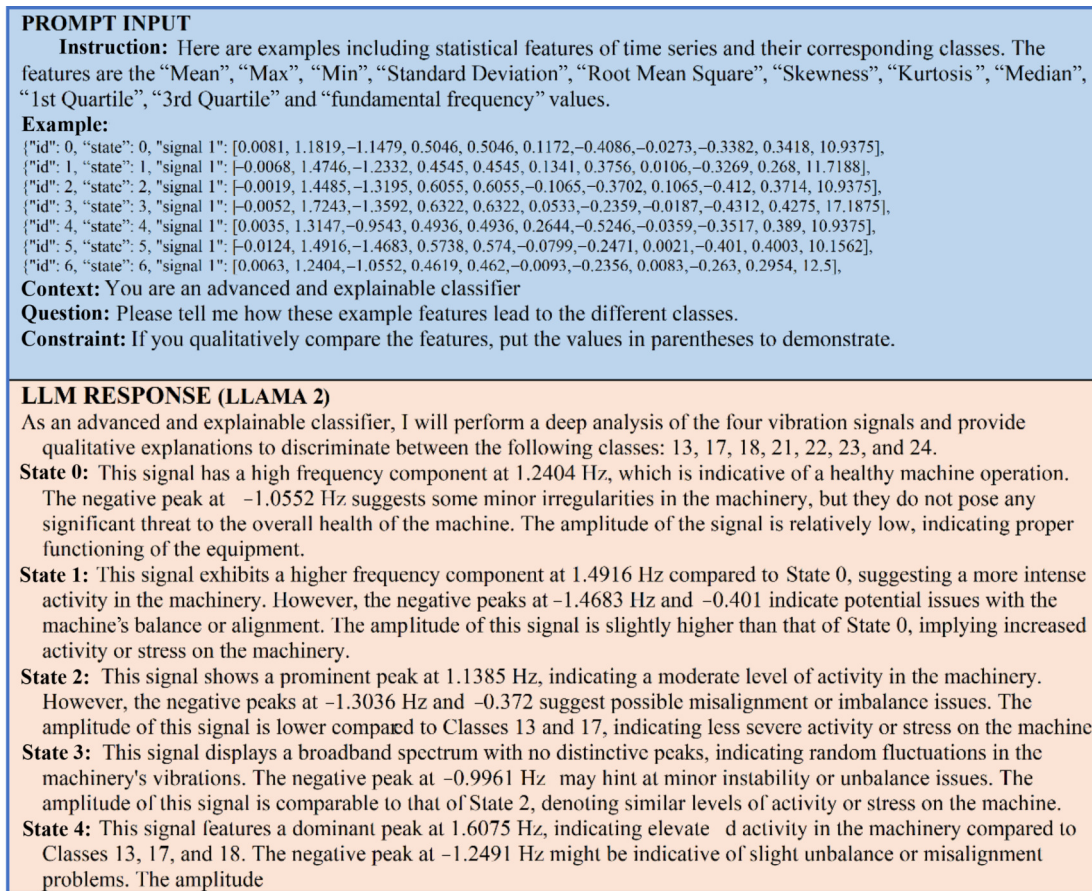


Fig. 16 Prompt and LLM’s corresponding responses for classifying structural states using extracted features of the laboratory frame structure using the LLAMA 2 model.

automating the preprocessing steps to provide general insights about data and building classifiers in a low-code manner. More specifically, scientists can provide step-by-step guidance for LLM-based models via CoT prompting strategies and then request the LLM to generate corresponding code snippets to perform these tasks.

4 Conclusions

In this study, a novel and practical SHM framework based on LLM and prompt engineering has been proposed and developed in a comprehensive fashion, including the overall workflow, background knowledge, technical know-how, and implementation details. The viability of the proposed method is investigated through realistic and publicly accessible experimental data; hence it is straightforward to reproduce the obtained results with the provided prompts. Based on the presented findings, the main conclusions can be summarized as follows.

1) A complete workflow for automatically performing SHM tasks using LLM and prompt engineering is proposed and implemented.

2) Directly utilizing LLMs to perform specific tasks such as detecting structural states, cannot provide

satisfactory results. However, employing LLMs in an indirect way, such as acting as an assistant to provide general insights about data and generating code snippets to perform SHM tasks, can achieve highly accurate results.

3) This pioneering work opens a new and fruitful avenue for future work in developing highly automated knowledge-informed LLM-assistant applications for SHM and other tasks in civil engineering because it liberates researchers and engineers from technical obstacles posed by programming languages with strict syntaxes or overly complicated mathematical formulations. Therefore, they can concentrate on core ideas, allocate more time and resources to experiment with different innovative ideas, and share their findings more straightforwardly with broader audiences because all framework steps are realized via human-machine interaction in natural languages.

4) The performance and applicability of the proposed method are quantitatively demonstrated via two examples, showing that with well-managed data and a proper flow of prompts, LLMs can well understand the problem and provide error-free ML-based classifiers that are capable of providing results with a high accuracy of up to more than 90% for the investigated SHM problems.

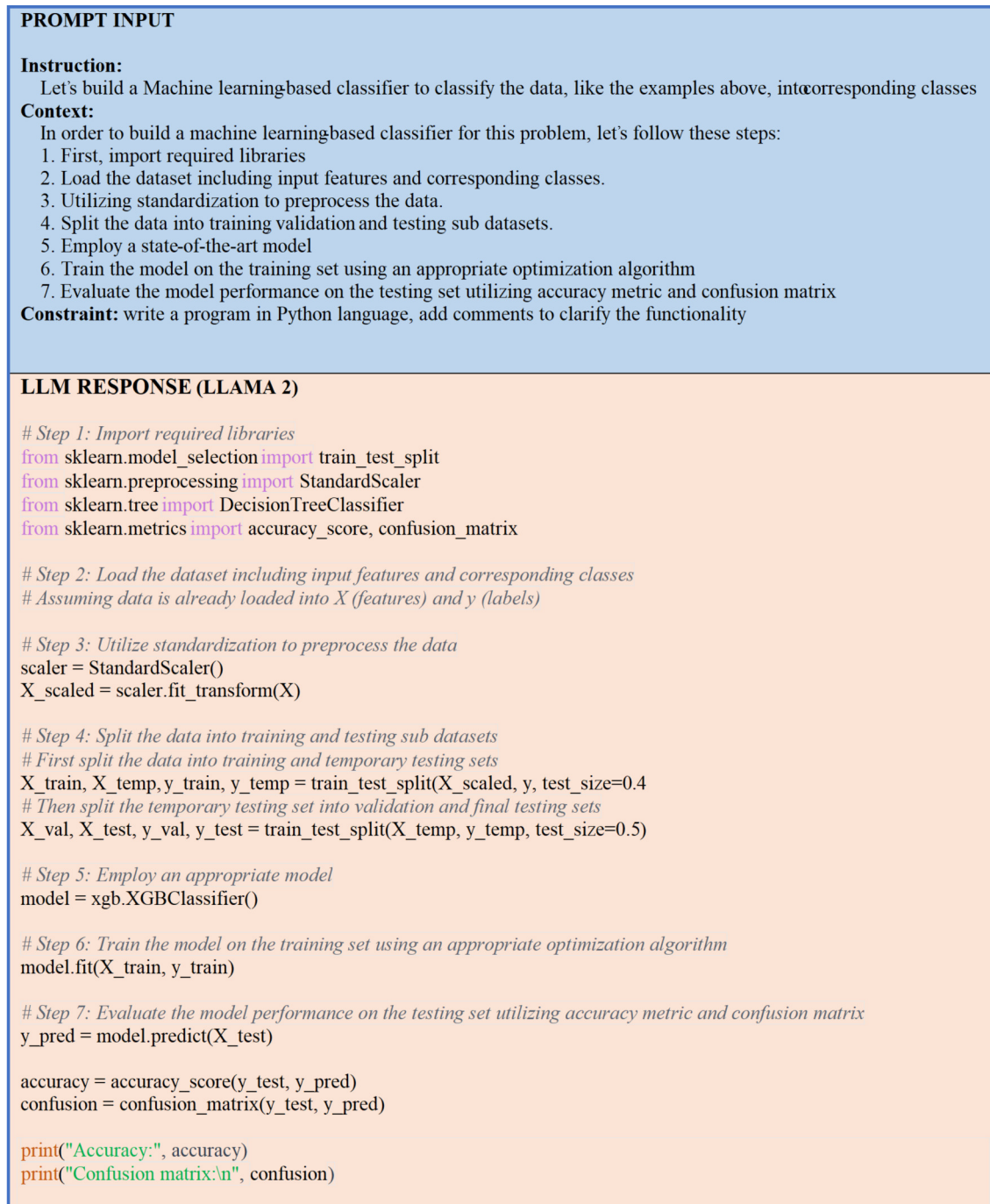


Fig. 17 Prompt and LLM's corresponding responses for building a ML-based classifier using the LLAMA 2 model.

The combination of LLM and ML algorithms may increase the black-box nature of the approach. Therefore, it is urged that the transparency of the LLM-assistant methods and the interpretability of their outputs should be improved. This can be achieved through a knowledge-informed prompt strategy and by including more contextual information in prompts. It is also noted that the current maximum prompt length is limited, but with increasing development in this active field, the next generation of LLM will certainly extend the prompt

length much longer; hence, we could incorporate more relevant instruction. Another limitation at this stage of the research is that a model train with vibration data generated by random excitation does not ensure performance well on other vibration data caused by earthquakes. This is because the statistical characteristics of the latter differ entirely from those of the formers in various perspectives, such as the frequency contents, the signal durations, and distribution shifts of amplitudes. To the best of the authors' knowledge, to build an AI model

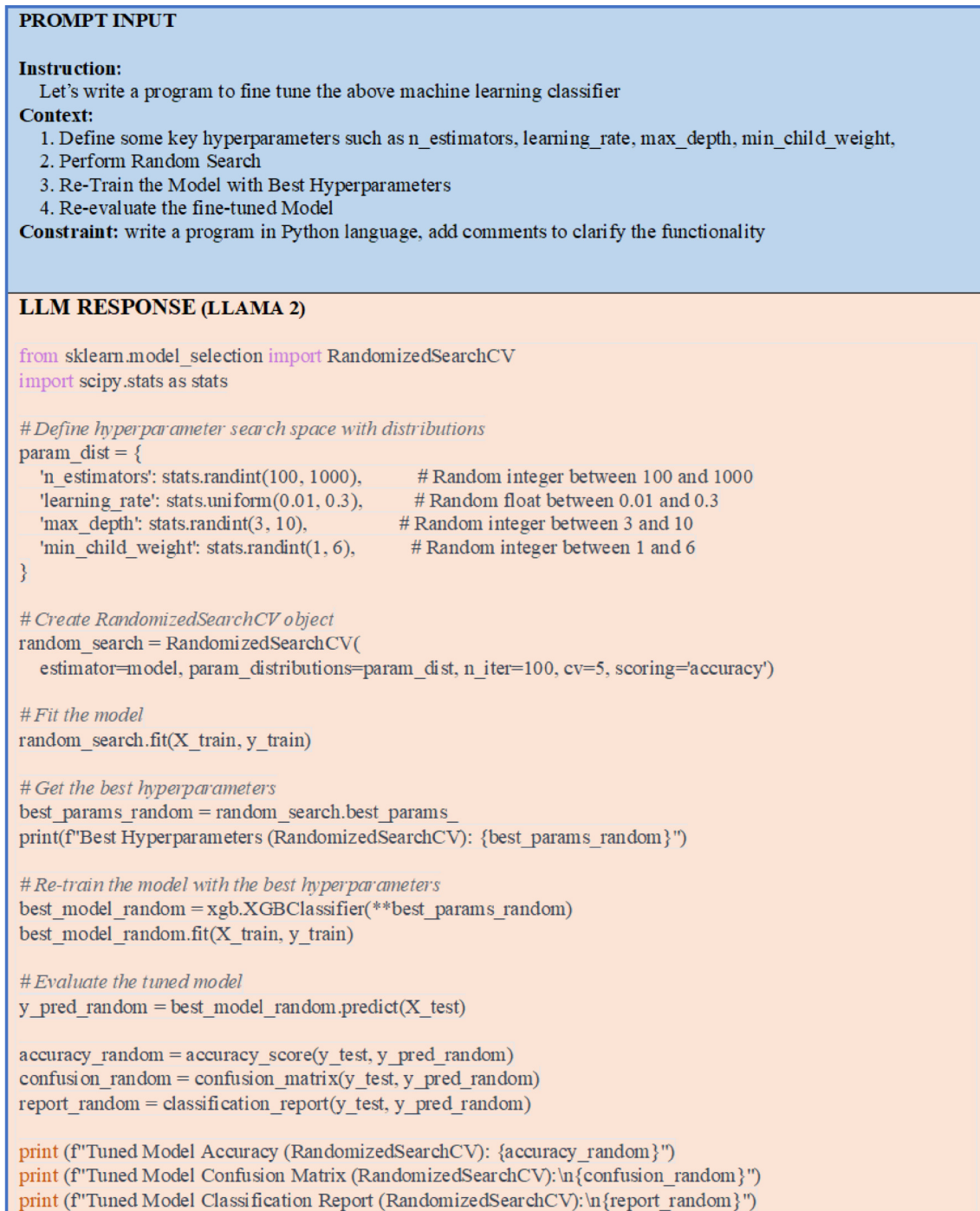


Fig. 18 Prompt and LLM's corresponding responses for fine-tuning a ML-based classifier.

Table 3 Results of structural damage detection using the classifier provided by LLM (LLAMA2) on the laboratory frame structure

Class	Precision	Recall	F1-score
0	0.89	0.85	0.87
1	0.89	0.87	0.88
2	0.92	0.97	0.94
3	0.88	0.90	0.89
4	0.96	0.96	0.96
5	0.90	0.90	0.90
6	0.86	0.86	0.86

that can perform on different vibration databases generated by random excitation, earthquake, or other types of excitations, we need to build a large-scale foundation model trained on massive, reliable data as done in computer vision [38], chemistry [39], Seismology [40].

For future research, it is interesting to extend the method to account for multi-modal input data like images and video in addition to vibration signals. For this purpose, it is necessary to engineer an adequate data

		Actual						
		State 0	State 1	State 2	State 3	State 4	State 5	State 6
Predicted	State 0	797	33	9	26	0	15	54
	State 1	34	907	42	24	1	13	22
	State 2	4	18	914	1	0	2	6
	State 3	7	22	5	899	34	8	19
	State 4	0	1	0	43	1008	1	2
	State 5	22	14	13	9	1	926	46
	State 6	36	28	8	21	1	60	910

Fig. 19 Detailed of structural damage detection results via confusion matrix.

PROMPT INPUT

Instruction: Write a program to plot the Receiver Operating Characteristic curves for the outputs of a classifier for the data like examples above.

Format: There are, in total, seven classes; let's plot the ROC curve for each class on a separate subplot.

Constraint: Add X label, Y label, and legend mentioning the area under the curve for each subplot.

Derived from LLM RESPONSE (GPT 3.5)

To plot Receiver Operating Characteristic (ROC) curves for each class, we first need to calculate the true positive rate (TPR) and false positive rate (FPR) for various thresholds. Then, we can plot these values to create the ROC curve. Below is a Python code to achieve this:

```
from sklearn.metrics import auc, roc_curve
from sklearn.metrics import RocCurveDisplay
import matplotlib.pyplot as plt

label_binarizer = LabelBinarizer().fit(y_train)
y_onehot = label_binarizer.transform(y_test)
y_score = model.predict_proba(X_test)
N_classes = len(np.unique(y_train))

fig, axs = plt.subplots(3,3, figsize=(12, 12))

for class_id in zip(N_classes):
    ax=axs[class_id // 3, class_id %3]
    RocCurveDisplay.from_predictions(y_onehot[:, class_id], y_score[:, class_id],
        name=f"ROC curve for state {class_id}",
        ax=ax)

    ax.plot([0,1],[0,1],':k', label='(AUC = 0.50)', linewidth=2)
    ax.set(xlabel="False Positive Rate", ylabel="True Positive Rate");

plt.tight_layout()
```

Fig. 20 Prompt and code snippet derived from LLM responses for plotting ROC curves using the GPT 3.5 model.

pipeline covering data management, data storage and date ingestion into LLMs. It is also necessary to fine-tune LLMs with appropriate data sets, although it is challenging for individuals to pre-train very big LLMs with personal computational devices. Another promising way is to link LLM with other tools and build a workflow where LLM serves as a central brain, sending requests to and receiving responses from task-specific tools, before

generating comprehensive answers to users. Finally, it is intriguing to deploy this framework for real structures in the next step of the study by supplement components such as a web app for visualizing results and an update procedure for accommodating new structural states or damage scenarios in addition to the presented inference ability of LLMs and domain knowledge from users' prompts.

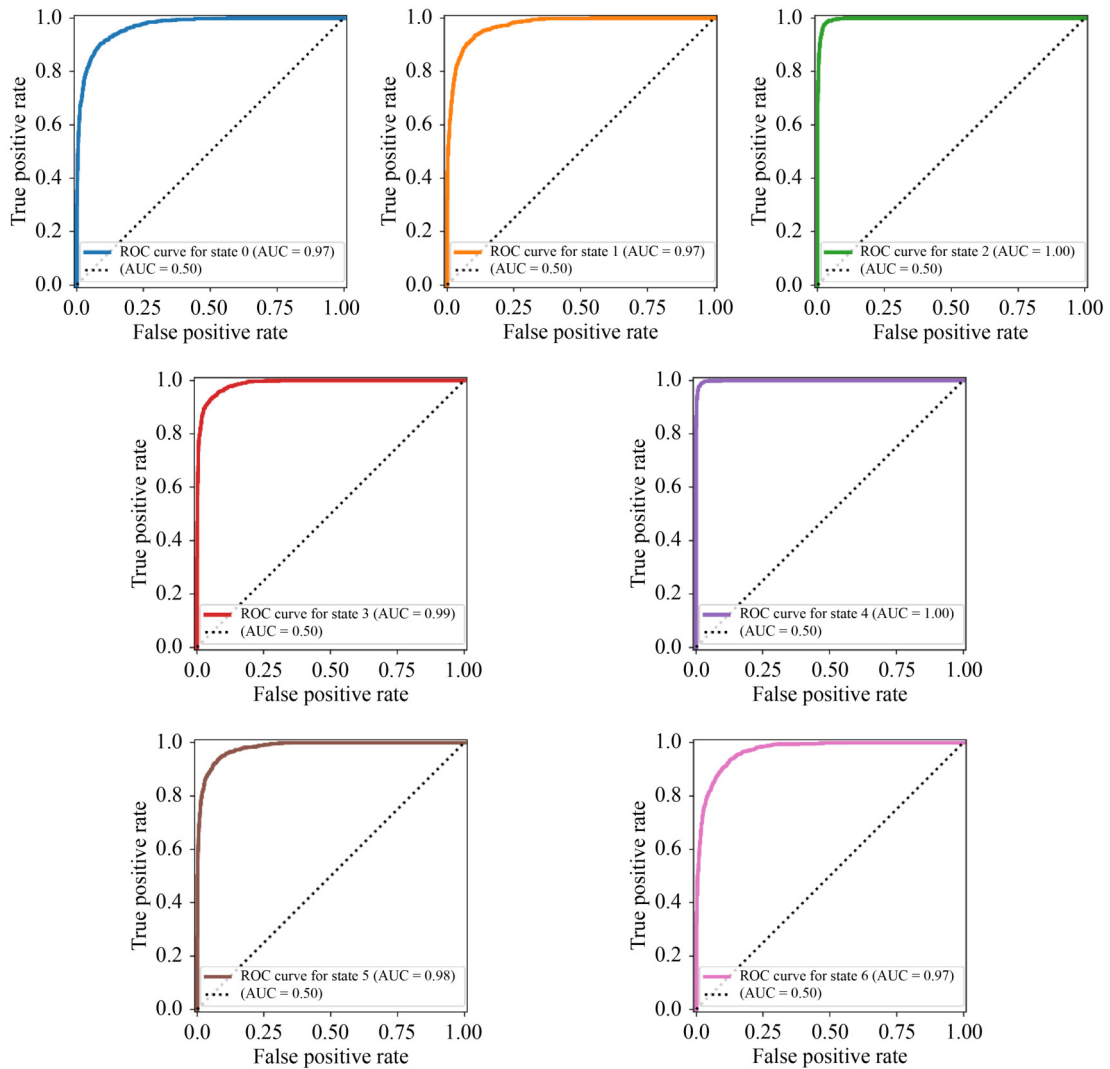


Fig. 21 Detailed of structural damage detection results via ROC curves.

Acknowledgements The research group of Development and Application of Advanced Materials and Modern Technologies in Construction would like to thank the Hanoi University of Civil Engineering for funding this research.

Competing interests The authors declare that they have no competing interests.

References

1. Christ M, Braun N, Neuffer J, Kempa-Liehr A W. Time series feature extraction on basis of scalable hypothesis tests (TSfresh—A python package). *Neurocomputing*, 2018, 307: 72–77
2. Dang H V, Raza M, Tran-Ngoc H, Bui-Tien T, Nguyen H X. Connection stiffness reduction analysis in steel bridge via deep CNN and modal experimental data. *Structural Engineering and Mechanics. International Journal*, 2021, 77(4): 495–508
3. Dang H, Nguyen T T. Robust vibration output-only structural health monitoring framework based on multi-modal feature fusion and self-learning. *Periodica Polytechnica. Civil Engineering*, 2023,

- 67(2): 416–430
4. Dang V H, Pham H A. Vibration-based building health monitoring using spatio-temporal learning model. *Engineering Applications of Artificial Intelligence*, 2023, 126: 106858
5. Lin S, Zheng H, Han B, Li Y, Han C, Li W. Comparative performance of eight ensemble learning approaches for the development of models of slope stability prediction. *Acta Geotechnica*, 2022, 17(4): 1477–1502
6. Guo H, Zhuang X, Rabczuk T. A deep collocation method for the bending analysis of Kirchhoff plate. 2021, arXiv: 2102.02617
7. Zhuang X, Guo H, Alajlan N, Zhu H, Rabczuk T. Deep autoencoder based energy method for the bending, vibration, and buckling analysis of Kirchhoff plates with transfer learning. *European Journal of Mechanics. A, Solids*, 2021, 87: 104225
8. Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh V M, Guo H, Hamdia K, Zhuang X, Rabczuk T. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 2020, 362: 112790
9. Guo H, Zhuang X, Chen P, Alajlan N, Rabczuk T. Stochastic deep

- collocation method based on neural architecture search and transfer learning for heterogeneous porous media. *Engineering with Computers*, 2022, 38(6): 5173–5198
10. Katz D M, Bommarito M J, Gao S, Arredondo P. GPT-4 passes the bar exam. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, 2024, 382: 257572753
 11. Strobelt H, Webson A, Sanh V, Hoover B, Beyer J, Pfister H, Rush A M. Interactive and visual prompt engineering for AD-HOC task adaptation with large language models. *IEEE Transactions on Visualization and Computer Graphics*, 2023, 29(1): 1146–1156
 12. Yong G, Jeon K, Gil D, Lee G. Prompt engineering for zero-shot and few-shot defect detection and classification using a visual-language pretrained model. *Computer-Aided Civil and Infrastructure Engineering*, 2023, 38(11): 1536–1554
 13. Lubiana T, Lopes R, Medeiros P, Silva J C, Goncalves A N A, Maracaja-Coutinho V, Nakaya H I. Ten quick tips for harnessing the power of ChatGPT in computational biology. *PLOS Computational Biology*, 2023, 19(9): e1011319
 14. Busch K, Rochlitzer A, Sola D, Leopold H. Just tell me: Prompt engineering in business process management. In: *Proceedings of International Conference on Business Process Modeling, Development and Support*. Cham: Springer Cham, 2023, 3–11
 15. Zhou K, Yang J, Loy C C, Liu Z. Learning to prompt for visionlanguage models. *International Journal of Computer Vision*, 2022, 130(9): 2337–2348
 16. Polak M P, Morgan D. Extracting accurate materials data from research papers with conversational language models and prompt engineering—Example of ChatGPT. 2023, arXiv: 2303.05352
 17. Lo A W, Singh M. *From ELIZA to ChatGPT: The Evolution of NLP and Financial Applications*. Cambridge, MA: MIT Libraries, 2023
 18. Wang J, Liu Z, Zhao L, Wu Z, Ma C, Yu S, Dai H, Yang Q, Liu Y, Zhang S, et al. Review of large vision models and visual prompt engineering. 2023, arXiv: 2307.00855
 19. Hatakeyama-Sato K, Yamane N, Igarashi Y, Nabae Y, Hayakawa T. Prompt engineering of GPT-4 for chemical research: What can/cannot be done. *Science and Technology of Advanced Materials: Methods*, 2023, 3(1): 226030
 20. Heston T F. Prompt engineering for students of medicine and their teachers. 2023, arXiv: 2308.11628
 21. Zhu J J, Jiang J, Yang M, Ren Z J. ChatGPT and environmental research. *Environmental Science & Technology*, 2023, 57(46): 17667–17670
 22. Neagu A. How can large language models and prompt engineering be leveraged in computer science education? Thesis for the Master's Degree. Delft: Delft University of Technology, 2023
 23. Peres R, Schreier M, Schweidel D, Sorescu A. On ChatGPT and beyond: How generative artificial intelligence may affect research, teaching, and practice. *International Journal of Research in Marketing*, 2023, 40(2): 269–275
 24. Xia Q, Maekawa T, Hara T. Unsupervised human activity recognition through two-stage prompting with ChatGPT. 2023, arXiv: 2306.02140
 25. Chopra A K. *Dynamics of Structures: Theory and Applications to Earthquake Engineering*. Upper Saddle River, NJ: Prentice Hall, 2006
 26. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E H, Le Q V, Zhou D. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 2022, 35: 24824–24837
 27. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L, Polosukhin I. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017: 6000–6010
 28. Chung H W, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, Li E, Wang X, Dehghani M, Brahma S, et al. Scaling instruction-finetuned language models. 2022, arXiv: 2210.11416
 29. Radford A, Narasimhan K, Salimans T, Sutskever I. Improving language understanding by generative pre-training. 2018. Available at the website of OpenAI
 30. Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, Bashlykov N, Batra S, Bhargava P, Bhosale S, et al. Llama 2: Open foundation and fine-tuned chat models. 2023, arXiv: 2307.09288
 31. Bernal S, Beck J, Ventura C. An experimental benchmark problem in structural health monitoring. In: *Proceedings of the Third International Workshop on Structural Health Monitoring*. Stanford, CA: IWSHM, 2001
 32. Ye X, Cao Y, Liu A, Wang X, Zhao Y, Hu N. Parallel convolutional neural network toward high efficiency and robust structural damage identification. *Structural Health Monitoring*, 2023, 22(6): 3805–3826
 33. Chi Y, Cai C, Ren J, Xue Y, Zhang N. Damage location diagnosis of frame structure based on wavelet denoising and convolution neural network implanted with inception module and LSTM. *Structural Health Monitoring*, 2024, 23(1): 57–76
 34. Dyke S J, Bernal D, Beck J, Ventura C. Experimental phase II of the structural health monitoring benchmark problem. In: *Proceedings of the 16th ASCE Engineering Mechanics Conference*. Reston, VA: ASCE, 2003
 35. Figueiredo E, Park G, Figueiras J, Farrar C, Worden K. *Structural Health Monitoring Algorithm Comparisons Using Standard Data Sets*. Technical Report LA-14393. 2009
 36. Hung D V, Hung H M, Anh P H, Thang N T. Structural damage detection using hybrid deep learning algorithm. *Journal of Science and Technology in Civil Engineering*, 2020, 14(2): 53–64
 37. Das S, Saha P, Patro S. Vibration-based damage detection techniques used for health monitoring of structures: A review. *Journal of Civil Structural Health Monitoring*, 2016, 6(3): 477–507
 38. Kirillov A, Mintun E, Ravi N, Mao H, Rolland C, Gustafson L, Xiao T, Whitehead S, Berg A C, Lo W Y, et al. Segment anything. In: *Proceedings of 2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris: IEEE, 2023
 39. Horawalavithana S, Ayton E, Sharma S, Howland S, Subramanian M, Vasquez S, Cosbey R, Glenski M, Volkova S. Foundation models of scientific knowledge for chemistry: Opportunities, challenges and lessons learned. In: *Proceedings of BigScience Episode #5—Workshop on Challenges & Perspectives in Creating Large Language Models*. Dublin: Association for Computational Linguistics, 2022, 160–172
 40. Si X, Wu X, Sheng H, Zhu J, Li Z. SeisCLIP: A seismology foundation model pre-trained by multi-modal data for multi-purpose seismic feature extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 2024, 62: 1–13