Multibody System Dynamics (2026) 67:1143–1170
https://doi.org/10.1007/s11044-026-10163-8

RESEARCH

Clearance size detection based on deep neural networks without feature extraction

Tien Vuong Nguyen[1] · Antonio J. Rodríguez[2] · Francisco González[2] · Aki Mikkola[1] · Jin-Gyun Kim[3] · Grzegorz Orzechowski[1]

Received: 25 November 2025 / Accepted: 13 April 2026 / Published online: 24 April 2026

Abstract

Clearance is an important phenomenon in mechanisms that stems from manufacturing imperfections and wear and tear. Undetected clearance can compromise machine operations, negatively impacting its performance, and cause premature damage that requires maintenance actions. Monitoring clearance growth is useful to improve maintenance plans and helps reduce the number of unwanted interruptions during machine operation. In this work, a prediction method that targets the determination of the clearance size based on minimal, raw sensor data and machine learning is proposed. The data in this study are generated using multibody dynamics simulations based on planar mechanisms and used to train and compare several types of neural networks in terms of their ability to assess clearance size. Results show that clearance parameters can be reliably estimated with appropriate combinations of sensor locations and type of neural network. The developed method offers reliable clearance detection based on measurements that can be obtained from physical systems and used to monitor the state of their clearance defects.

✉ T.V. Nguyen
vuong.nguyen@lut.fi

A.J. Rodríguez
antonio.rodriguez.gonzalez@udc.es

F. González
f.gonzalez@udc.es

A. Mikkola
aki.mikkola@lut.fi

J.-G. Kim
jingyun.kim@khu.ac.kr

G. Orzechowski
grzegorz.orzechowski@lut.fi

[1] Department of Mechanical Engineering, LUT University, Yliopistonkatu 34, Lappeenranta, 53850, Finland

[2] Laboratorio de Ingeniería Mecánica, CITENI, Universidade da Coruña, Industrial Campus, Ferrol, 15403, Spain

[3] Department of Mechanical Engineering (Integrated Engineering), Kyung Hee University, Yongin-si, 17104, Gyeonggi-do, Republic of Korea

Keywords Clearance estimation · Neural networks · Raw sensors data · Multibody dynamics

1 Introduction

Clearance refers to the small gap or free motion that exists between two connected components in a mechanical system. In machines, it typically appears in joints, bearings, or other connections where parts are not in continuous tight contact. As a result, the presence of clearance allows additional movement that may involve impacts and intermittent contact between the components. This leads to vibration and noise, reduces overall machine performance, and increases heat generation and wear. Over time, excessive clearance can cause damage to components and eventually lead to premature system failure.

In traditional multibody system analysis, a joint is typically assumed to be an ideal connection between bodies, without any clearance or flexibility. This assumption simplifies the modeling process but does not always represent real mechanical systems accurately. In practice, joints often exhibit small gaps or elastic deformations that affect the relative motion between components. Neglecting these effects can lead to discrepancies between simulated and measured results, particularly in systems subjected to dynamic loads or wear [1].

The simulation of joints affected by clearance has attracted the interest of the multibody dynamics community for years, e.g., [2, 3]. A great deal of the effort invested by the research community has been geared towards developing appropriate representations of the dynamics of mechanical systems affected by this defect. This is a challenging task because the resulting motion and forces depend on a large number of factors that include the geometry of the joint, clearance size, contact stiffness, link flexibility in the mechanical system, and the existence and characteristics of friction and lubrication at the contact area.

Clearance size has been identified as a main factor in the dynamic response of systems affected by the defect. For small defects, it has little impact on the time-history of the overall system motion, but introduces important modifications in the accelerations, which stem from the impact forces between the bodies in contact at the affected joint [4]. It also plays a critical role in the identification of the stage in which the defect is, because clearances are often caused by wear, and their size increases over time, as a result of the defective mechanical behavior of the affected system [5, 6]. Larger clearance gaps give rise to greater vibration responses. In industrial machinery, this is associated with a deterioration of machine performance; the increase in the defect size eventually reaches a point in which normal operation is no longer possible.

Conventional approaches to deal with this joint clearance include reactive and scheduled maintenance [7]. Reactive approaches consist in repairing the machine when the defect has already caused a failure that prevents operation. Scheduled maintenance carries out this task according to a schedule, usually regardless of the internal state of the machine. In contrast to these solutions, predictive maintenance relies on information from the device to determine the appropriate time to intervene, decreasing the downtime of the machinery and its associated costs. To be effective, predictive maintenance needs accurate data about the machine status. The monitoring and characterization of clearance size are necessary tools for this. Often, however, these tasks need to be performed using data from a reduced set of sensors, which is usually limited because industrial machinery does not include extensive sensor arrays.

Over the past few years, machine learning (ML) has been proven to be a powerful tool for fault detection and diagnosis in engineering systems. Unlike model-based techniques

that rely upon explicit physical equations, ML models are capable of identifying the underlying patterns directly from high-dimensional data, especially in the case of complicated and nonlinear input-output mappings. Numerous ML algorithms, ranging from shallow methods such as Support Vector Machines [8], Decision Trees [9], and Random Forests (RF) [10], to deeper architectures such as Neural Networks [11], have been successfully integrated to detect anomalous behaviors. Shallow algorithms are often used for problems with limited data or relatively simple relationships, while deep neural networks can capture more complex patterns and handle high-dimensional or sequential data, such as time series sensor measurements. Besides fault detection, ML algorithms are increasingly integrated into predictive maintenance and monitoring frameworks, where the objective is not only to detect faults but also to forecast their progression. For example, Bayesian deep learning models have been used to predict the remaining useful life of industrial components and parts to provide uncertainty-based probability predictions in favor of informed maintenance decisions [12]. Additionally, algorithms such as XGBoost and Random Forest can be used in manufacturing production lines to process real-time IoT sensor readings to detect warning signs of impending failures [13]. These forecasting attributes allow maintenance work to be done at the right time, minimize unexpected shutdowns, and improve maintenance scheduling, all factors essential for sustaining the reliability and efficiency of advanced engineering systems.

The advantages of ML are also recognized and applied in the field of multibody system dynamics, where traditional modeling can be extremely complex due to nonlinear kinematic and kinetic effects, high degrees of freedom, and coupled interactions. Traditional model-based approaches rely on explicit physical equations that struggle with the highly nonlinear behavior of certain components, which are difficult to model accurately. The finite element method is an example. These methods often require simplifying assumptions or computationally expensive iterative solvers, making them unsuitable for real-time applications. Various researchers have applied ML algorithms to a variety of tasks, such as replacing rigid multibody system dynamic simulations [14], longitudinal prediction of vehicle dynamics [15], real-time musculoskeletal prediction [16], or friction force prediction for hydraulic actuators [17]. Furthermore, data-driven methods are particularly suited for systems with unknown, impractical, or computationally intractable governing functions, as shown by their use in soft robotics for modeling the complex, nonlinear behavior of compliant materials [18]. These studies demonstrate that ML can serve as a powerful alternative to traditional approaches by offering accurate predictions and faster computation without requiring explicit mathematical formulations. Furthermore, ML techniques offer the flexibility to generalize to new scenarios not explicitly simulated, a capability particularly valuable for safety-critical, performance-oriented, and industrial applications.

Neural networks have been used in the past in the context of clearance simulation to describe joint dynamics [19]. Although many research studies have been conducted on joint clearance, the majority of them focus on modeling techniques [20–22], analyzing the effects of clearance on revolute and translational joints [23], contact force modeling [24], wear analysis [25], and dynamic interaction [26]. These approaches provide a foundation and critical insights into clearance, but are often limited by computationally intensive iterative solvers, resulting in slow computing speed. In addition, since they usually depend on simplifying assumptions to reduce complexity, the results might be less realistic and difficult to apply to real-world scenarios. However, despite extensive research on modeling and analysis, the use of data-driven methods for clearance prediction remains largely unexplored. The data-driven approach, which learns the underlying pattern directly from sensor measurement data, provides the ability to overcome the limitations of traditional physical or

mathematical-based methods. This helps to reduce the reliance on assumptions and accelerate computation speed. Moreover, this kind of method is suitable for real-time monitoring and maintenance, where early detection of clearance characteristics is essential in enhancing the system reliability and preventing unexpected failures.

In this paper, a data-driven approach to determine clearance size in mechanical systems is proposed. The introduced approach is based on neural networks and was developed with the purpose of predicting the magnitude of a clearance when its location is known or assumed, using accelerometer data as input. In order to capture the nonlinearity of the sensor data, several algorithms are used, ranging from shallow ML techniques such as Random Forest (RF) or Extreme Gradient Boosting (XGBoost), to more complex ones such as Time-Delay Neural Network (TDNN) and Recurrent Neural Networks (RNNs), including long short-term memory (LSTM) and gated recurrent unit (GRU). Training data were generated by means of the forward-dynamics simulation of the motion of a 2D slider-crank mechanism with attached accelerometer sensors. The performance of trained data-driven models was demonstrated through numerical examples. The main contribution of this paper is determining the suitability of leveraging ML algorithms to characterize clearances in mechanical systems directly from sensor readings, assessing the most appropriate methods to deal with this problem. The impact of the number of sensors and their location on the reliability of the prediction algorithms developed was also investigated.

2 Numerical model and data production

This work uses synthetic data from a multibody system (MBS) dynamics simulation environment to train the neural network. Multibody dynamics is a powerful tool for the analysis and design of mechanical systems [27], which considers them composed of bodies (rigid or flexible) connected through joints that allow a certain motion. The effect of clearances on mechanisms has been an object of study in the multibody community during the last decades [1]. The approaches to describe the phenomena caused by clearances can be classified in two categories: penalty methods [28] and non-smooth dynamics [29]. The penalty approach, often used to address clearance problems, is based on Hertz's theory of deformable bodies and assumes very small deformations during contact. Several contact models can be used with the penalty approach, such as Hunt and Crossley's [30] and Lankarani and Nikravesh's [31].

This work focuses on developing a neural network to extract information about clearances in planar mechanisms from the readings of sensors mounted on the affected device. Since this study is done in a simulation environment, the multibody model has to include a contact model to characterize clearances. It should be remarked that the priority of the work is not to study which contact model is the most accurate. Instead, a model is selected that can represent the contact dynamics with a certain level of realism. Previous research [32] confirms that the selection of one or another contact model does not result in critical differences in the behavior of the studied physical system, although it does cause differences in the magnitude of the contact force. The Lankarani and Nikravesh contact model [31] was, accordingly, selected in this study. This contact model computes the contact force as

$$F_N = K\delta^n \left[1 + \frac{3(1-e^2)}{4}\frac{\dot{\delta}}{\dot{\delta}^{(-)}}\right] \tag{1}$$

where F_N is the contact force in the normal direction of the contact surface, K is the contact stiffness which depends on the contact properties (material, geometry, etc.), e is the

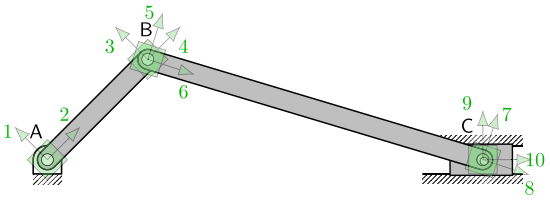


Fig. 1 Slider-crank model employed in this work with all the sensors installed. The details of each sensor are available in Table 3 (Color figure online)

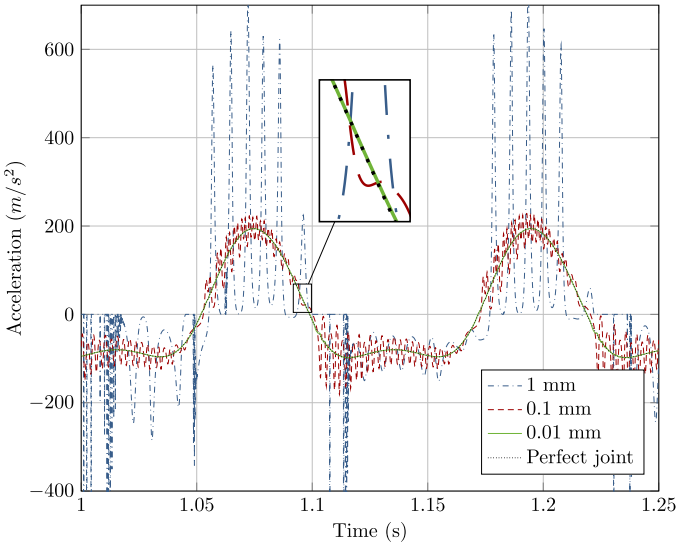


Fig. 2 Slider acceleration for different clearance sizes in the slider-crank mechanism employed in this work. The clearance is located in the slider (joint C) and the crank is rotating at a constant speed of 500 rpm (Color figure online)

coefficient of restitution, δ is the relative penetration or indentation between the bodies in contact, and $\dot{\delta}^{(-)}$ is the initial impact velocity, which remains constant until the contact is finished and the clearance enters free-flight mode. Further details regarding the calculation of additional contact parameters can be found in [31, 33].

The test problem under study is a slider-crank mechanism (Fig. 1), used in multiple machines in industry, which is modeled in this work as a planar linkage. To illustrate the effect that the clearance has on this type of mechanism, Fig. 2 shows the horizontal acceleration of the slider under different clearance sizes. It can be seen that the magnitude of the impacts increases with clearance size, with a strong influence on the acceleration profile of the slider.

The example is modeled using a set of n natural coordinates $\mathbf{q}$ related by m kinematic constraint equations $\mathbf{\Phi} = \mathbf{0}$. The dynamics of the system can be described as

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{Q} \tag{2}$$

$$\mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0} \tag{3}$$

where $\mathbf{M}$ is the mass matrix of the system, $\mathbf{\Phi}_{\mathbf{q}} = \partial\mathbf{\Phi}/\partial\mathbf{q}$ is the Jacobian matrix of the constraints, $\boldsymbol{\lambda}$ is a set of m Lagrange multipliers, and $\mathbf{Q}$ includes the applied forces [34].

The set of equations described in (2) and (3) is a system of second-order differential-algebraic equations (DAE). To solve a system of this type, a multibody dynamics formulation is required. In this work, the R-matrix method [34] has been selected. The R-matrix method defines the following velocity transformation,

$$\dot{\mathbf{q}} = \mathbf{R}\dot{\mathbf{z}} \tag{4}$$

where $\mathbf{z}$ stands for a set of independent coordinates, of size equal to the number of degrees of freedom of the mechanism, and $\mathbf{R}$ is the corresponding velocity transformation matrix, which verifies $\mathbf{R}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathbf{q}} = \mathbf{0}$. The derivative with respect to time of equation (4) is

$$\ddot{\mathbf{q}} = \mathbf{R}\ddot{\mathbf{z}} + \dot{\mathbf{R}}\dot{\mathbf{z}} \tag{5}$$

Equation (5) can be substituted in the dynamics equations (2) to obtain

$$\mathbf{R}^{\mathrm{T}}\mathbf{M}\mathbf{R}\ddot{\mathbf{z}} = \mathbf{R}\left(\mathbf{Q} - \mathbf{M}\dot{\mathbf{R}}\dot{\mathbf{z}}\right) \tag{6}$$

which is a system of second-order ordinary differential equations (ODE). Equation (6) can be integrated by means of conventional numerical formulas. In this work, the semi-implicit forward-Euler integration formula is used:

$$\dot{\mathbf{q}}_k = \dot{\mathbf{q}}_{k-1} + h\ddot{\mathbf{q}}_{k-1} \tag{7}$$

$$\mathbf{q}_k = \mathbf{q}_{k-1} + h\dot{\mathbf{q}}_k \tag{8}$$

where h is the integration step-size. Since contact problems are stiff, a small integration step-size is required to deal with them. The numerical experiments conducted in this work were integrated with a step-size $h = 10^{-5}$ s, which delivered a reasonable computational cost; decreasing the step-size below this value did not result in noticeable modifications in the computed dynamics. For more precise simulations, e.g., in cases with faster dynamics, it could be convenient to use a variable-step integration formula, so that the step-size is further reduced during the impact phase if necessary [33].

In this workcase, the mechanism is considered to have only one clearance at a time, which can be present between the crank and the connecting rod (point B) or the connecting rod and the slider (point C). The slider-crank mechanism is moving on a horizontal plane, without gravity effects, at a constant angular speed of the crank. The total simulated time is 3 seconds. The simulation starts with 0 velocity and accelerates until the target velocity during the first 0.5 seconds. Then, the velocity is controlled by a PID that determines the required torque in the crank (point A) to keep the speed constant. A total of 1000 datasets have been generated, each one with different clearance radii and velocities. The clearance radii were selected following a random distribution between a minimum value of 0 and a maximum value of 10^{-3} m. The angular velocity of the crank varies from 500 rpm to 1000 rpm, following a random distribution. The initial conditions do not vary between datasets.

The properties of the slider-crank used are summarized in Table 1 and the contact parameters are shown in Table 2. Note that the radial clearance is obtained by varying the bearing diameter and keeping the diameter of the journal constant.

The mechanism can also mount multiple sensors. In this particular case, the use of accelerometers is considered in order to measure accelerations in the longitudinal and perpendicular directions of each element of the mechanism, yielding a total of 10 measurements. Accelerometers are the most suitable sensor for clearance detection, since the effect of impacts in the joints is most relevant at the acceleration level, as shown in Fig. 2.

Table 1 Properties of the slider-crank mechanism considered in this work

	Mass (kg)	Inertia (kg·m^2)	Length (m)
Bar A-B	0.3	1×10^{-5}	0.05
Bar B-C	0.21	2.5×10^{-4}	0.12
Slider	0.14	1.05×10^{-5}	0.015

Table 2 Parameters of the contact model considered in this work

Restitution coefficient	0.9
Young's modulus	200 GPa
Poisson's ratio	0.3
Journal radius	0.01 mm

Table 3 List of the acceleration measurements available in the mechanism. The ID corresponds with the number assigned in Fig. 1

ID	Body	Measurement
1	Crank	Acceleration perpendicular to the crank (A)
2	Crank	Acceleration in the longitudinal direction of the crank (A)
3	Crank	Acceleration perpendicular to the crank (B)
4	Crank	Acceleration in the longitudinal direction of the crank (B)
5	Connecting rod	Acceleration perpendicular to the connecting rod (B)
6	Connecting rod	Acceleration in the longitudinal direction of the connecting rod (B)
7	Connecting rod	Acceleration perpendicular to the connecting rod (C)
8	Connecting rod	Acceleration in the longitudinal direction of the connecting rod (C)
9	Slider	Vertical acceleration of the slider
10	Slider	Horizontal acceleration of the slider

3 Prediction models

This section describes not only the shallow machine learning algorithms but also the neural network models considered and compared in this research. Throughout this section, multivariate time-series input data from sensor measurements are assumed

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n], \tag{9}$$

where $\mathbf{x}_t$ $(t = [1, \ldots, n])$ is a vector containing sensor measurements at time t. Our goal is to predict the clearance radius $r \in [0, 10^{-3}]$ m. The challenge here is capturing the nonlinear relationships between sensor readings and clearance size, requiring methods that can handle both temporal dependencies and complex feature interactions.

Shallow ML algorithms such as RF and XGBoost were first used as baselines, as they are widely adopted in fault detection and regression tasks and provide a reference point for comparison. The study was then extended to the sequence learning approach: TDNN to handle short-term temporal patterns from sensor data, LSTM, and GRU to capture long-term nonlinear relationships.

3.1 Shallow algorithms

Shallow machine learning algorithms, while not able to process complex problems, such as image or sequence processing, are still used due to their robustness, interpretability, and effectiveness in small or moderate-sized datasets. Among these, tree-based ensemble techniques are considered to be more advanced and powerful than others and are widely used. They rely on ensembles of relatively simple decision trees and do not exploit temporal or hierarchical features directly. RF constructs multiple decision trees in parallel using bootstrap aggregation and combines their outputs through majority voting or averaging, resulting in reduced variance and improved generalization [35]. To be more specific, RF builds an ensemble prediction

$$\hat{\mathbf{y}} = \frac{1}{B} \sum_{i=1}^{B} T_i(\mathbf{x}), \tag{10}$$

where B is the total number of trees, each tree T_i processes the flattened feature vector, in other words, it is the prediction from the $i-th$ individual decision tree.

Additionally, XGBoost builds trees sequentially using gradient boosting, of which each new tree corrects the residual errors of the previous ensemble [36]

$$\hat{\mathbf{y}} = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \tag{11}$$

where K is the total of boosting trees, $f_k(\boldsymbol{x}_i)$ is the contribution from the $k-th$ tree to predicting the $i-th$ sequence.

3.2 Neural networks

Given that the data generated in Sect. 2 is a time series, more complex algorithms should be considered in order to capture the temporal patterns and map them to clearance sizes. This is when neural networks, including TDNN, LSTM, and GRU, come in to handle this phenomenon.

3.2.1 Time-delay neural network

Feed Forward Neural Network (FFNN) is the simplest type of neural network; simply put, it is the fundamental type of artificial neural network. In this architecture, data moves in one direction only, from the input layer, through one or many hidden layers, and finally to the output layer. This one-way flow of information is what gives the network its "feedforward" characteristic. The structure of these networks is relatively simple, making them a foundational model for understanding more complex neural network architectures [37].

Time-Delay Neural Network (TDNN) is a specialized version of an FFNN designed to handle sequential or time series data, originally introduced for modeling temporal dependencies in speech signals. It operates by using a sliding window that scans through the data, allowing it to detect local patterns and features, regardless of their position in the sequence [38].

A key feature of TDNNs is that the same weights are applied across different time steps [38]. This weight sharing is similar to the functionality of convolutional filters in a Convolutional Neural Network (CNN) [39], enabling the network to learn a single set of feature

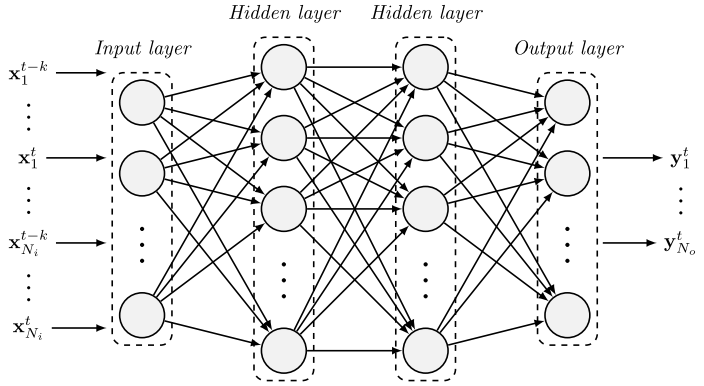


Fig. 3 Architecture of a TDNN

detectors that are effective regardless of where the pattern occurs in the input sequence. This not only reduces the number of parameters to be learned, but also makes the network robust to temporal translations of the patterns it detects.

In a TDNN, each hidden unit at time t receives inputs not only from the current input vector $\mathbf{x}_t$ but also from a fixed number of past input vectors $(\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-K})$. The hidden activation is defined as

$$\mathbf{h}_t = \sigma\left(\sum_{k=0}^{K} \mathbf{W}_k \mathbf{x}_{t-k} + \mathbf{b}\right), \tag{12}$$

where $\mathbf{x}_t \in \mathbb{R}^d$ is the input feature vector at time t, $d = N_i$ is the number of features, K is the sliding window size, $\mathbf{W}_k \in \mathbb{R}^{h \times d}$ is the learnable weight matrix for time delay k, h is the number of hidden units, $\mathbf{b} \in R^h$ is the bias, $\sigma(\cdot)$ is the activation function, and the network outputs a number of values N_o.

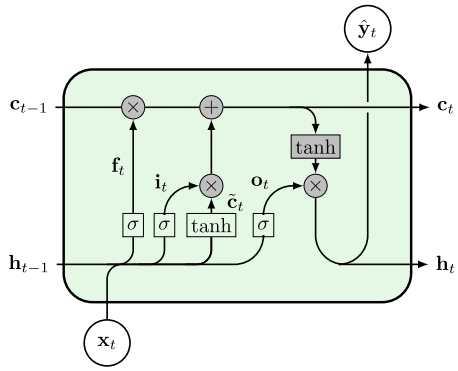
The output of each hidden neuron is then fed into subsequent layers (Fig. 3), which integrate the information from multiple time steps to form a more global understanding of the sequence. This hierarchical processing allows the network to build a representation of the data that captures both local details and broader temporal context.

3.2.2 Mechanism of an LSTM network

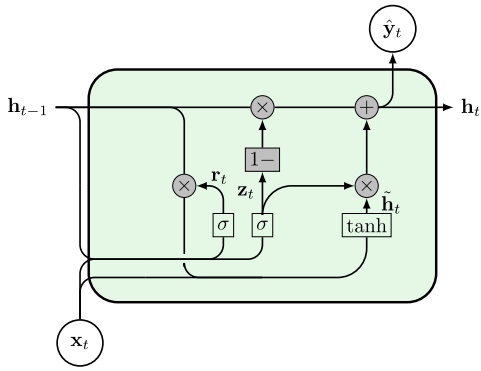
Long Short-Term Memory (LSTM) is a type of Recurrent Neural Networks (RNNs) and was proposed by [40]. LSTM was designed to overcome long-term dependency problems in RNNs that are affected by the problems of the vanishing gradient [41].

An LSTM network typically consists of multiple memory cells (one for each time step in a sequence) within a structure that includes various components to regulate information flow. Let $\mathbf{W}_*, \mathbf{U}_*, \mathbf{b}_*$ be trainable parameters, σ be the sigmoid activation function, and $\odot$ be elementwise product, for each LSTM unit or cell (Fig. 4a), it consists of the following key components. The forget gate $\mathbf{f}_t$ removes unnecessary information received from the cell state

$$\mathbf{f}_t = \sigma(\mathbf{W}_\mathbf{f}\mathbf{x}_t + \mathbf{U}_\mathbf{f}\mathbf{h}_{t-1} + \mathbf{b}_\mathbf{f}). \tag{13}$$



(a) Diagram of an LSTM memory cell.



(b) Diagram of a GRU cell.

Fig. 4 Comparison of LSTM and GRU cells (Color figure online)

The input gate $\mathbf{i}_t$ selects which information is needed to be added to the cell state

$$\tilde{\mathbf{c}}_\mathbf{t} = \tanh(\mathbf{W}_{\tilde{\mathbf{c}}}\mathbf{x}_t + \mathbf{U}_{\tilde{\mathbf{c}}}\mathbf{h}_{t-1} + \mathbf{b}_{\tilde{\mathbf{c}}}), \tag{14}$$

$$\mathbf{i}_t = \tanh(\mathbf{W}_\mathbf{i}\mathbf{x}_t + \mathbf{U}_\mathbf{i}\mathbf{h}_{t-1} + \mathbf{b}_\mathbf{i}). \tag{15}$$

The cell state $\mathbf{c}_t$ represents the "memory" of the LSTM, which carries information through time steps

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t. \tag{16}$$

The output gate $\mathbf{o}_t$ determines which information from the cell state is used as output

$$\mathbf{o}_t = \sigma(\mathbf{W}_\mathbf{o}\mathbf{x}_t + \mathbf{U}_\mathbf{o}\mathbf{h}_{t-1} + \mathbf{b}_\mathbf{o}). \tag{17}$$

Finally, the hidden state $\mathbf{h}_t$ is the output of each LSTM unit at each time step, derived from the cell state

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \tag{18}$$

3.2.3 Mechanism of a GRU network

Similar to LSTM, Gated Recurrent Unit (GRU), is an alternative to RNNs and was proposed by [42]. However, it was designed to be less complex and faster than LSTM.

Let $\mathbf{W}_*, \mathbf{U}_*, \mathbf{b}_*$ be trainable parameters, σ be the sigmoid activation function, and $\odot$ be the elementwise product, the main components of a GRU, shown in Fig. 4b are as follows. The reset gate $\mathbf{r}_t$ determines how much of the previous information should be forgotten

$$\mathbf{r}_t = \sigma(\mathbf{W}_\mathbf{r}\mathbf{x}_t + \mathbf{U}_\mathbf{r}\mathbf{h}_{t-1} + \mathbf{b}_\mathbf{r}). \tag{19}$$

The update gate $\mathbf{z}_t$ combines the functionality of an LSTM's forget and input gates, determining how much of the previous information should be used

$$\mathbf{z}_t = \sigma(\mathbf{W}_\mathbf{z}\mathbf{x}_t + \mathbf{U}_\mathbf{z}\mathbf{h}_{t-1} + \mathbf{b}_\mathbf{z}). \tag{20}$$

Unlike LSTM, GRU does not have a separate cell state. Instead, the hidden state $\mathbf{h}_t$ directly serves as both memory and the output of the unit at each time step. It carries information forward through time steps, incorporating both new input and relevant past information

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_\mathbf{h}\mathbf{x}_t + \mathbf{U}_\mathbf{h}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_\mathbf{h}), \tag{21}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t. \tag{22}$$

4 Numerical experiments

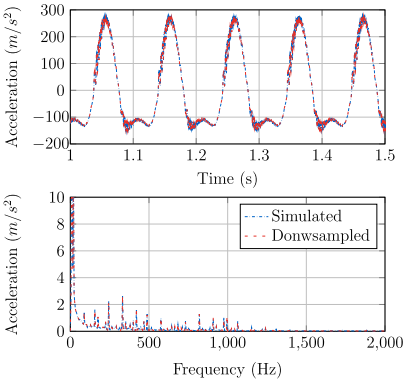
4.1 Data generation and characteristics

The dataset utilized in this study consists of raw time series simulations, each containing accelerometer readings collected over time. Additionally, they were generated from a high-fidelity multibody system dynamics environment, as stated in detail in Sect. 2. In these simulations, clearance is defined as small, non-contacting gaps within a joint, particularly between the connecting rod and the slider (joint C in Fig. 1), and also explored at joint B in specific configurations. The presence of clearance leads to characteristic impacts and vibrations as illustrated in Fig. 2. The simulations recorded accelerometer sensor data across various operating conditions, specifically varying the clearance size between 0 and 10^{-3} m, and the crank's angular velocity between 500 and 1000 rpm. This resulted in a total of 1000 unique simulations. Each of them provided 10 raw measurements of acceleration in different positions and directions of the mechanism (Fig. 1).

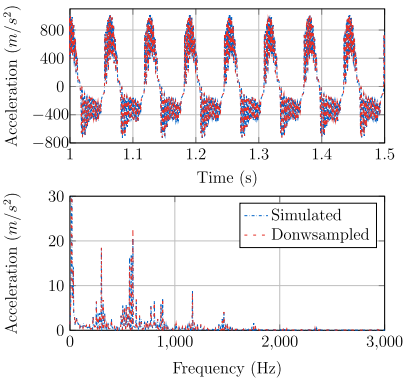
4.2 Data preparation and preprocessing

These simulations underwent several preprocessing steps before being used for training. To focus on the stable operational phase of the mechanism and exclude initial transient behavior, the raw sensor data were initially truncated, with processing beginning from 200 000th data point among 300 000. Subsequently, the data were downsampled by a factor of 10, meaning that every 10th data point from the truncated series was selected. While this step fundamentally reduced the computational load for processing large time series datasets, the main motivation for this specific downsampling factor was its demonstrated important benefit on model performance.

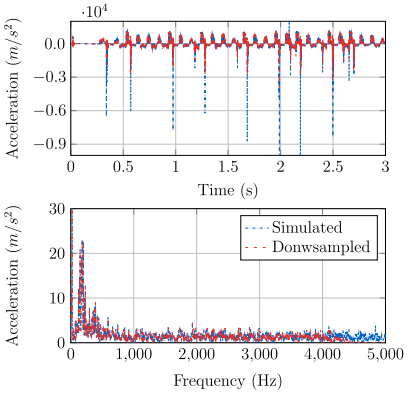
In addition, real-world sensors cannot capture accelerations at 100 kHz. Commercial sensors, such as [43], can gather measurements up to frequencies of around 10 kHz, which reinforces the need to downsample the data to get a more realistic operational scenario. To simulate a real sensor behavior, the downsampling was made by decimating the data



(a) Clearance of 0.051 mm and rotation speed of 582.5 rpms.



(b) Clearance of 0.203 mm and rotation speed of 946.5 rpms.



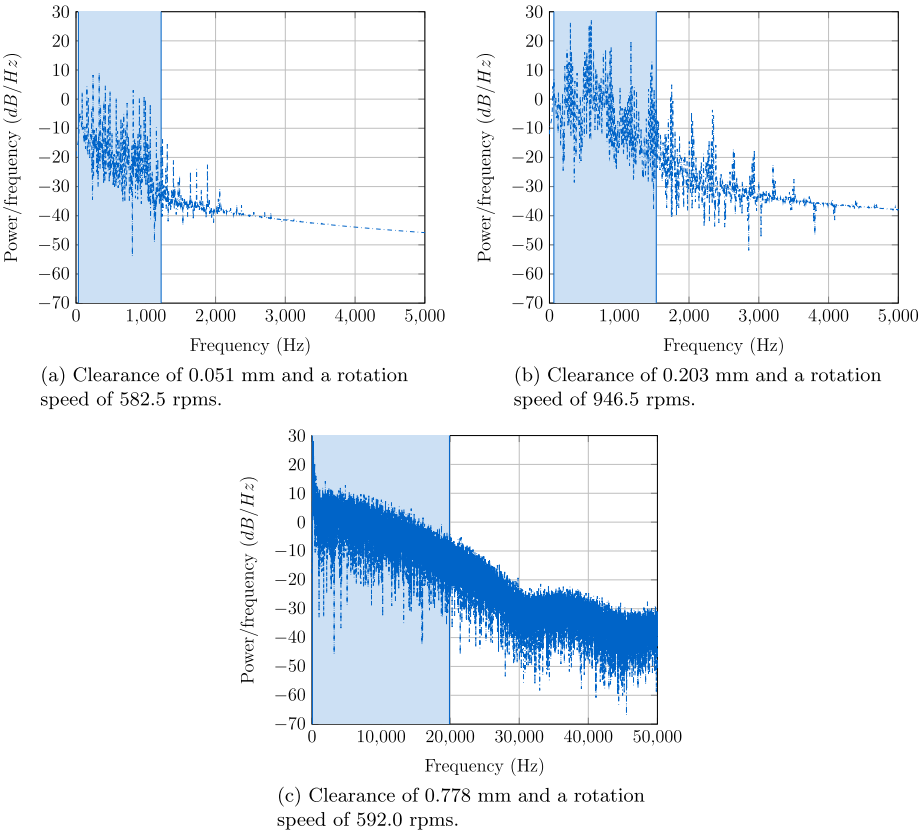
(c) Clearance of 0.778 mm and rotation speed of 592.0 rpms.

Fig. 5 Comparison of the acceleration signals measured by sensor 10 during the simulation and the downsampled version used in the NN expressed in time-domain and frequency-domain (Color figure online)

and using a lowpass Chebyshev Type I infinite impulse response (IIR) filter of order 8 before downsampling. This data processing implies the loss of information in the acceleration measurements.

To assess the effects that downsampling can introduce in the neural network predictions, the acceleration signals were analyzed in the frequency domain. Fig. 5 shows the time-domain and frequency-domain representation of the accelerations measured by sensor 10 under three different clearances located in joint C. It can be seen that most of the information related to the vibrations introduced by the clearance is below 1 kHz. For large clearances, the time-domain representation shows that the magnitudes of stronger impacts are not fully captured after downsampling the data. These impacts are also seen in the FFT as components of the acceleration at high frequencies, which are not captured after downsampling the data.

A more quantitative point of view can be obtained by calculating the frequency at which the 99% of the signal bandwidth is occupied (Fig. 6). This is done by computing a periodogram power spectral density. The occupied bandwidth is the difference in frequency between the point where the integrated power crosses 0.5% and 99.5% of the total power in



(a) Clearance of 0.051 mm and a rotation speed of 582.5 rpms.

(b) Clearance of 0.203 mm and a rotation speed of 946.5 rpms.

(c) Clearance of 0.778 mm and a rotation speed of 592.0 rpms.

Fig. 6 Occupied bandwidth of 99% for the signals shown in Fig. 5. The blue zone represents the area where the 99% of the bandwidth is contained (Color figure online)

the spectrum. To focus the analysis on the vibration effects, frequencies below 50 Hz were removed, allowing for the capture only of the accelerations derived from the presence of clearances. For low- and medium-size clearances, a sampling frequency between 1 kHz and 2 kHz would be enough for capturing almost all the information of the vibrations. For large clearances, close to 20 kHz is required. This, in concordance with the comparison made in Fig. 5, indicates that less accuracy when estimating large clearances is expected.

Then, feature selection was performed. Distinct accelerometer measurements from Table 3 were used as input for the models. Depending on the considered measurements, different cases are observed and their performances are shown in Sect. 5. Each input sequence to the models had a fixed length of 200 steps. This means that, for each prediction, the model considered 200 consecutive downsampled time steps of sensor readings. This sequence length was determined during preliminary experiments. A shorter sequence length, such as 100, was found to result in degraded model performance. The 200-time-step window, on the other hand, allowed the models to effectively learn the temporal patterns. A total of 10 000 data points will be obtained if a simulation with 100 000 data points is downsampled by a factor of 10. By sliding a window of 200 time steps across this data, the models can produce multiple predictions. In particular, the number of predictions per simulation is $\frac{10\,000}{200} = 50$. Additionally, if d denotes the number of sensors used, the input data in Equa-

Table 4 Hyperparameter configuration of shallow models

	RF	XGBoost
Number of trees	100	300
Max depth	20	20
Learning rate	-	0.01

tion (9) is

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{200}], \qquad \mathbf{x}_t \in \mathbb{R}^d. \tag{23}$$

The clearance under investigation is specifically located between the connecting rod and the slider (joint C in Fig. 1). For each experimental run, acceleration measurements, as detailed in Table 3, were continuously collected. From this comprehensive set of sensors, different measurements from Table 3 were specifically selected to serve as inputs to our models. It is important to note that the data was generated from the environment simulation as described in Sect. 2, where only accelerometers were installed and their readings recorded as output. Therefore, this study focuses on clearance detection utilizing this specific sensor modality.

The entire dataset was randomly split on the simulation level into training, validation, and testing sets with proportions of 70%, 15%, and 15%, respectively, using three different random seeds (42, 98, and 1024) to ensure that the models' performance is not dependent on a specific data distribution. Finally, normalization was applied. All input features were normalized using *StandardScaler* fitted exclusively on the training data. This ensures that the mean of the training data is 0 and the standard deviation is 1, preventing features with larger scales from dominating the learning process. The same scaler was then applied to the validation and test sets. The target labels, or clearance sizes, were scaled using a *MinMaxScaler*, also fitted only on the training labels. This transforms the labels to a range between 0 and 1, which can improve training stability. The inverse transform was applied to predictions for evaluation in the original scale.

4.3 Shallow models

To provide baselines and offer a comparative perspective on the efficiency of different model complexities, shallow machine learning algorithms, including RF and XGBoost, were used. They are widely recognized for their simplicity, interpretability, and computational efficiency, especially when compared to more complex recurrent neural networks designed for temporal data. For these shallow models, the input, which consists of 200 timesteps across selected accelerometer measurements, was flattened into a single, high-dimensional feature vector. For instance, if four measurement signals were used, each data point became an 800-element vector (200 timesteps * 4 signals). This flattening process, although it helped to convert the data into a suitable format for RF and XGBoost, discarded the explicit temporal dependencies and sequential order within the sensor readings. Models were then tasked with patterns from these concatenated features without knowledge of their original temporal relationship.

The implementation was based on `scikit-learn` and `XGBoost` libraries. Regarding hyperparameters, as detailed in Table 4, the RF model included 100 decision trees, each limited to the maximum depth of 20 to control model complexity. In addition, the XGBoost model used the same number of input features as the RF one. The model was trained

with 300 boosting rounds, with each tree constrained to a maximum depth of 20 for regularization. A learning rate of 0.01 was applied to control gradient updates. These choices were made to ensure robust baseline performance without extensive optimization, reflecting a focus on comparing architecture paradigms rather than achieving peak performance for shallow models.

4.4 Neural network architectures

Our neural network models, including TDNN, LSTM, and GRU architectures, were trained to predict clearance size from time series sensor data. The architectures were implemented in Pytorch with the size of input depending on the number of selected accelerometer features. The layers used a hidden size of 256. For depth, all three models consisted of three stacked layers.

4.4.1 Training process

The models were trained using PyTorch on available GPU hardware if present, otherwise on the CPU. Training was performed with Adam optimizer [44], chosen for its adaptive learning rate capabilities and proven effectiveness in optimizing deep neural networks. A learning rate of 0.0001 was used, a common starting point for Adam. The Mean Squared Error (MSE) loss function was chosen as the criterion for training, which is standard for regression tasks. MSE is particularly suitable here as it strongly penalizes larger prediction errors, which is critical for accurately predicting clearance size. During training, data were processed in batches of 64 samples, and each model was trained for a maximum of 200 epochs. It allowed for effective parallel processing on the GPU hardware, hence accelerating training iterations. To improve training stability and prevent exploding gradients, gradient clipping with a maximum norm of 10 was applied during backpropagation for both LSTM and GRU models. Additionally, to optimize training time, an early stopping mechanism was implemented, terminating training if the validation loss did not improve for 20 consecutive epochs.

The training environment was strictly controlled to ensure full reproducibility across all experiments. Global seeds were set using `random.seed(SEED)`, `numpy.random.seed(SEED)`, and `torch.manual_seed(SEED)` before any random operations, including data splitting, weight initialization, and batch shuffling. To guarantee deterministic behavior on the GPUs, CUDA determinism was enforced by setting `torch.backends.cudnn.deterministic = True` and `torch.backends.cudnn.benchmark = False`. Furthermore, the Pytorch Dataloader utilized the global RNG to ensure consistent shuffling reproducibility.

Regarding the checkpoint strategy, the model weights from the best epoch were restored for final evaluation to ensure the models were compared at their peak performance.

4.4.2 Sequence modeling architectures

The TDNN takes a different approach by treating the sensor sequences as signals to be analyzed using convolutional operations; in other words, the TDNN applies 1-dimensional convolutions across the temporal dimension. The input data maintains the same structure of 200 timesteps with a specific number of sensor readings per step. The architecture consists of three convolutional layers, each with 256 filters and a sliding window with a size of 5, which slides across the time series to detect local temporal patterns. Each convolutional layer

is followed by a rectified linear unit (ReLU) activation function [45] to introduce nonlinearity. After the convolutional processing, three different aggregation strategies were evaluated: flattening, global average pooling, and global max pooling. TDNN with flattening approach concatenates all feature channels across the temporal dimension and processes them through two fully connected layers (512 neurons with ReLU, then output with sigmoid). Additionally, the global max pooling approach extracts the maximum activation across time for each of the 256 channels, while the global average pooling approach computes the mean activation across time for each channel. The pooling variants use a single fully connected layer with sigmoid activation for the final prediction. Each strategy was evaluated to determine the optimal architecture, as detailed in Sect. 5.

In both the LSTM and GRU models, the input data consists of sequences of sensor data readings taken over time. Each sequence is made up of 200 timesteps, and at each step, the system records a number of different acceleration measurements. Rather than processing one sequence at a time, the model is trained using batches of 64 sequences to make training more efficient. Both of these models then read through each sequence step by step, from the beginning to the end, building an internal memory of the temporal patterns. In the LSTM model, this internal memory uses two stacked layers of 256 neurons each, with separate cell states for long-term memory and hidden states for short-term focus. Additionally, GRU simplifies this architecture by using three layers of 256 neurons with a single hidden state that continuously updates. After processing all 200 timesteps, both models extract the final hidden state as a 256-dimensional summary vector, which is then passed through a fully connected layer to predict the clearance size.

4.5 Evaluation process

To ensure a fair and consistent comparison, a unified evaluation process was applied to both shallow and neural network models. After the training phases, the performance of each model was tested on the unseen data, comprising 15% of the total generated data. For every sample in the test set, regardless of the model type, the preprocessed sensor data were fed as input. Each model then produced a single predicted output value, representing its estimation of the clearance radius. This predicted clearance value was directly compared against the known true clearance size for that specific simulation.

The accuracy of these estimations was quantified by directly comparing the predicted values with the actual clearance values using standard metrics. These metrics provide a thorough evaluation of prediction accuracy, Mean Absolute Error (MAE) quantifies the average magnitude of errors, and Root Mean Squared Error (RMSE) provides the square root of the average squared errors. The selection of metrics allowed for a proper observation and understanding of model performance. However, since the clearance sizes are significantly small (in the range of 0 and 10^{-3} m), it would be difficult to interpret. Instead, normalized MAE (NMAE) and normalized RMSE (NRMSE) by range were used in order to provide measures of the error relative to the total clearance change, thus improving the interpretability of model performance.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{N} |\hat{y}_i - y_i|}{N}, \qquad \mathrm{NMAE} = \frac{\mathrm{MAE}}{y_{\max} - y_{\min}} \times 100\%, \tag{24}$$

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{N}}, \qquad \mathrm{NRMSE} = \frac{\mathrm{RMSE}}{y_{\max} - y_{\min}} \times 100\%. \tag{25}$$

To assess the distribution and robustness of prediction errors, two additional metrics were integrated. Median Absolute Error (Median AE) provides a measure of typical error that is robust to outliers, representing the middle value of all absolute errors. The 95th Percentile Absolute Error (95th PAE) indicates the worst-case error for 95% of predictions, effectively characterizing the upper bound of error distribution while excluding extreme outliers:

$$\text{Median AE} = \text{median}(|\hat{y}_i - y_i|), \tag{26}$$

$$\text{95th PAE} = P_{95}(|\hat{y}_i - y_i|), \tag{27}$$

where P_{95} denotes the 95th percentile operator.

Furthermore, to assess model robustness against different initialization conditions, all three models were each evaluated across three different train-validation-test splits (42, 98, and 1024) and 10 different weight initializations per split (42, 98, 123, 456, 789, 1024, 2048, 4096, 8192, 9999). This resulted in 30 training runs per model. For each model, performance metrics were aggregated across these 30 runs and reported as mean $\pm$ standard deviation. Additionally, 95% confidence intervals (95% CI) were computed to provide a range estimate of the true mean performance with 95% confidence. Statistical significance between models was assessed using paired t-tests with a significance level of $\alpha = 0.01$.

4.6 Cross-model validation with perturbed data

To perform cross-model validation, the trained models were evaluated on perturbed test data, including errors in parameters whose value is uncertain, such as those used in the contact model. These perturbations simulate variations in the physical system's behavior and modeling choices, testing whether the trained neural networks can generalize beyond the specific simulation conditions used during training. The same evaluation metrics described in Sect. 4.5 were applied to assess the model's performance under each perturbation scenario.

Five different perturbations were introduced, and new sets of 165 simulations were executed with random error level for validation purposes. First, regarding the restitution coefficient e (Equation (1)), which is common to most of the Hertzian contact models and directly affects contact damping, a level of uncertainty is expected as its value is case-specific. While training data were generated with a restitution coefficient of 0.9, an error of +10% and -20% was introduced for the new validation set to avoid exceeding the upper limit value of 1. Second, the contact stiffness K (Equation (1)) was addressed. Since the contact force is proportional to this value, and its computation depends on multiple parameters, an error in between -20% and +20% was introduced to generate a new validation set. Third, the choice of contact model was considered, as different Hertzian contact models can result in varying force magnitudes that affect contact dynamics [32]. To cover this uncertainty, the Hunt and Crossley model [30] was selected to generate an additional validation set. Furthermore, zero-mean Gaussian noise ($\alpha = 1\%$ of sensor signal standard deviation) was added to the test sensor data in original physical units before rescaling. This simulates realistic measurement uncertainties while maintaining consistency with the training data normalization. Finally, a consolidated validation data set was generated, including all described perturbations, with the exception of noise.

5 Results

This section presents results from different shallow machine learning and neural network models. All experiments were performed on a workstation featuring two AMD EPYC 7713

Table 5 Performance metrics of shallow machine learning models at joint C

Model	NMAE (%)	NRMSE (%)	Training time (s)	Evaluation time (s)
RF	6.19%	8.82%	55.03	0.15
XGBoost	9.3%	11.94%	49.88	0.01

64-Core Processors, 512 GB of RAM, and two NVIDIA A100 80 GB PCIe GPUs. The computational environment was built on Python 3.10.12, utilizing PyTorch 2.5.1 and CUDA 12.8.

To ensure robust evaluation and account for both data-dependent and initialization-dependent variability, each deep learning model was evaluated across 30 independent runs following the methodology described in 4.5. For visual presentation of representative predictions, the splitting seed was set to 98, and the weight initialization seed was set to 42 to ensure reproducibility throughout this section.

5.1 Models evaluation with baseline sensor configuration

In this specific case, 4 acceleration measurements with IDs 5, 6, 7, and 8 from Table 3 were chosen as inputs while the clearance was located between the slider and connecting rod (joint C in Fig. 1) for all models. This selection served as a starting point for assessing all models' performance, with a more comprehensive sensor sensitivity analysis conducted in Sect. 5.4 to investigate the impact of various sensor configurations and identify optimal sensor sets.

5.1.1 Shallow machine learning models

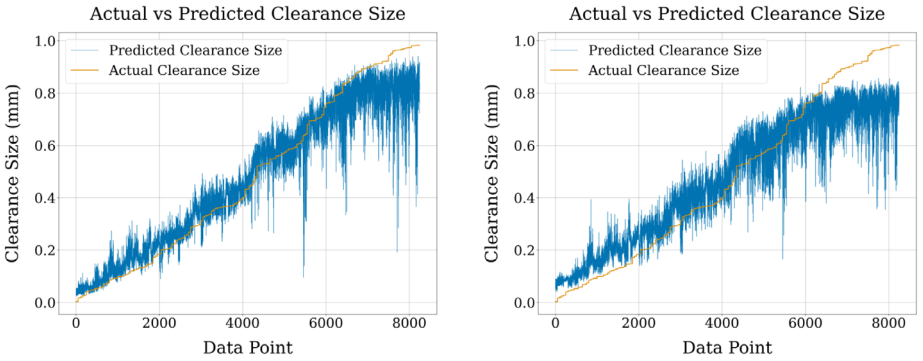
Starting with shallow machine learning models by implementing RF and XGBoost as initial approaches, baseline performance benchmarks were quickly established. From a computational perspective, these shallow models enabled rapid prototyping and experimentation with minimal cost, as can be seen in Table 5. In addition, as shown in Figs. 7a and 7b, although RF and XGBoost models can capture the overall trend of the predictions, they still fail to provide sufficiently accurate results.

Furthermore, in both Figs. 7a and 7b, there are noticeable deviations from actual values, especially at higher clearance sizes. This is as expected since these shallow algorithms do not have the ability to handle the temporal data as efficiently as the neural network ones. Instead, they process flattened feature vectors, thereby losing explicit temporal dependencies.

5.1.2 Deep learning models

For deep learning models, three architectures were evaluated: TDNN, LSTM, and GRU.

Prior to comparative analysis, three TDNN variants with different aggregation strategies were investigated: concatenating all time-step features (flattening), global max pooling, and global average pooling. Each TDNN variant was trained across 30 independent runs. The flattening strategy achieved the best overall performance with NMAE of $1.42 \pm 0.14\%$ and NRMSE of $2.15 \pm 0.23\%$. In comparison, TDNN with global max pooling yielded NMAE of $1.62 \pm 0.33\%$ and NRMSE of $2.42 \pm 0.41\%$, while global average pooling resulted in NMAE of $2.39 \pm 0.45\%$ and NRMSE of $3.33 \pm 0.53\%$. Subsequently, only the flattening-based TDNN is reported in the comparative analysis.



(a) Performance of the RF model.

(b) Performance of the XGBoost model.

Fig. 7 Comparison of RF and XGBoost model performances (Color figure online)

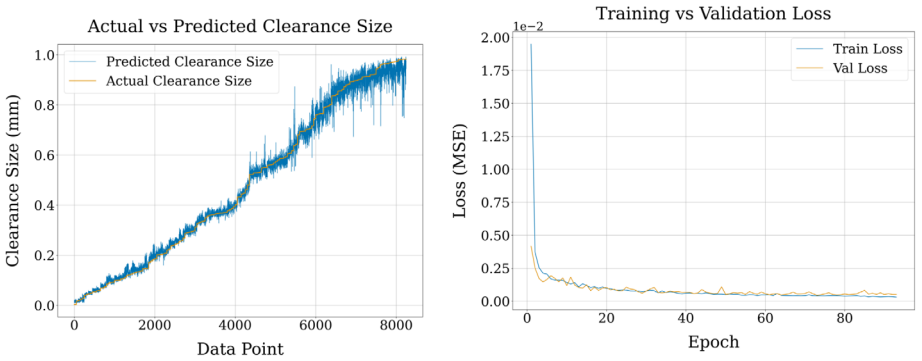
Table 6 Performance metrics of deep learning models at joint C

Model	NMAE (%)	NRMSE (%)	Training time (s)	Evaluation time (s)
TDNN	1.43 ± 0.19	2.19 ± 0.27	259.47 ± 48.53	0.19 ± 0.001
LSTM	0.49 ± 0.14	0.80 ± 0.19	1369 ± 380.6	0.89 ± 0.02
GRU	0.47 ± 0.18	0.74 ± 0.26	1439.3 ± 372	0.89 ± 0.05

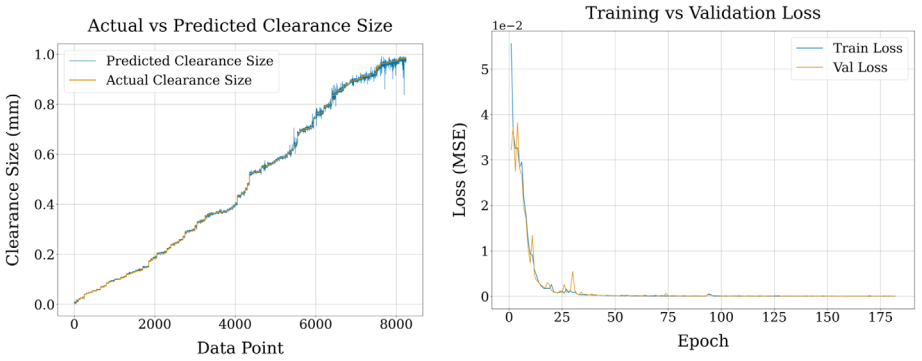
As can be seen from Table 6, the performance evaluation across 30 runs revealed that LSTM and GRU achieved comparable results with no statistically significant difference (paired t-test, $p > 0.01$ for all metrics). Specifically, LSTM achieved an NMAE of $0.49 \pm 0.14\%$ (95% CI: [0.44%, 0.54%]) and NRMSE of $0.8 \pm 0.19\%$ (95% CI: [0.73%, 0.87%]), while GRU obtained an NMAE of $0.47 \pm 0.18\%$ (95% CI: [0.4%, 0.54%]) and NRMSE of $0.74 \pm 0.26\%$ (95% CI: [0.64%, 0.84%]). GRU demonstrated marginally lower error metrics, though the difference was not statistically significant. In contrast, TDNN with flattening showed higher errors with NMAE of $1.43 \pm 0.19\%$ (95% CI: [1.36%, 1.5%]) and NRMSE of $2.19 \pm 0.27\%$ (95% CI: [2.09%, 2.29%]), which were statistically significantly worse than both LSTM and GRU (paired t-test, $p < 0.001$). However, TDNN offered a considerable computational advantage, with an average training time of 259.47 ± 48.53 seconds and evaluation time of 0.19 ± 0.001 seconds, compared to LSTM (1369.03 ± 380.6 seconds training, 0.89 ± 0.02 seconds evaluation) and GRU (1439.3 ± 372 seconds training, 0.89 ± 0.05 seconds evaluation). This represents approximately an 81% reduction in training time and 78% reduction in evaluation time.

Figure 8 shows representative predictions from a single run (data split seed 98, weight initialization seed 42) for each neural network architecture, demonstrating typical prediction quality.

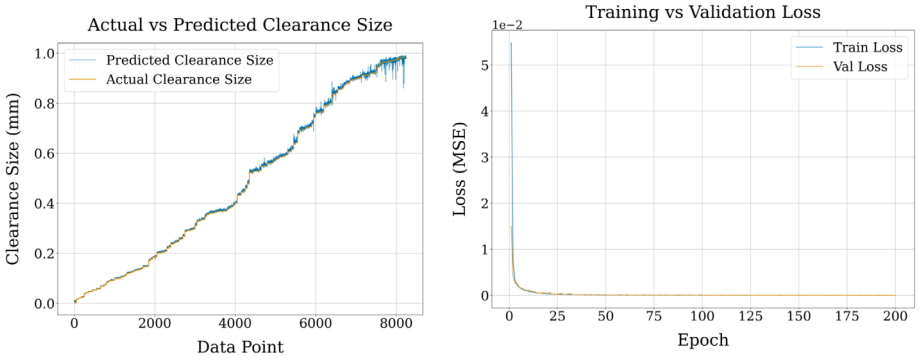
Both LSTM and GRU models deliver precise and stable predictions, characterized by the tightest fit to actual values with minimal visible noise for larger clearances. Additionally, TDNN successfully captures the overall trend but shows higher variance, indicating less refined individual data point estimations. This is understandable since GRU and LSTM models capture long-term dependencies using memory cells and gates, while TDNN's convolutional approach for local patterns is effective but may not capture global temporal context as effectively as RNNs.



(a) TDNN.



(b) LSTM.



(c) GRU.

Fig. 8 Performance comparison between true and predicted clearance values of neural network models using 4 sensors with clearance in joint C: (a) TDNN, (b) LSTM, and (c) GRU (Color figure online)

From a qualitative point of view, the effect that the reference clearance size has on the estimations is also interesting: the noise in the estimations increases with the clearance size. According to [2], when the radial clearance increases, the number of impacts and the maximum impact acceleration increase as well. The motion inside the clearance becomes more chaotic and unpredictable. However, when the clearance size is reduced, the dynamics are smoother since the journal and the bearing are in contact for longer periods of time. This,

Table 7 Model performance across perturbation scenarios

Model	Metric	Base	Restit.	Stiff.	Hunt-C	All	Noise
LSTM	NMAE (%)	0.49 ± 0.14	0.55 ± 0.16	0.49 ± 0.15	0.54 ± 0.16	0.51 ± 0.15	0.5 ± 0.14
	NRMSE (%)	0.8 ± 0.19	1.48 ± 0.44	0.95 ± 0.36	1.07 ± 0.32	0.84 ± 0.19	0.82 ± 0.18
	Med. AE (μm)	3.43 ± 1.23	3.76 ± 1.3	3.52 ± 1.28	3.82 ± 1.35	3.56 ± 1.28	3.49 ± 1.2
	95% PAE (μm)	13.6 ± 3.75	14.3 ± 4.17	12.92 ± 3.86	14.38 ± 4.35	13.78 ± 4.17	14.1 ± 3.57
GRU	NMAE (%)	0.47 ± 0.18	0.54 ± 0.21	0.47 ± 0.19	0.52 ± 0.21	0.49 ± 0.2	0.49 ± 0.18
	NRMSE (%)	0.74 ± 0.26	1.44 ± 0.43	0.93 ± 0.38	0.97 ± 0.37	0.79 ± 0.26	0.77 ± 0.25
	Med. AE (μm)	3.28 ± 1.49	3.66 ± 1.81	3.36 ± 1.58	3.68 ± 1.8	3.47 ± 1.68	3.4 ± 1.45
	95% PAE (μm)	13.1 ± 5.03	13.96 ± 5.37	12.79 ± 5.08	13.86 ± 5.35	13.56 ± 5.49	13.9 ± 4.94
TDNN	NMAE (%)	1.43 ± 0.19	1.61 ± 0.17	1.42 ± 0.17	1.64 ± 0.17	1.48 ± 0.16	1.43 ± 0.19
	NRMSE (%)	2.19 ± 0.27	3.21 ± 0.15	2.41 ± 0.2	2.86 ± 0.17	2.21 ± 0.17	2.19 ± 0.27
	Med. AE (μm)	9.55 ± 1.52	10.5 ± 1.52	9.67 ± 1.58	11 ± 1.61	10.32 ± 1.49	9.56 ± 1.53
	95% PAE (μm)	42.6 ± 4.99	46.09 ± 3.85	40.68 ± 3.81	47.14 ± 3.64	42.6 ± 3.6	42.77 ± 5.02

together with the effects of the sensors' downsampling explained in Sect. 4.2, explains why the method provides lower accuracy when the mechanism presents large clearances.

5.2 Evaluating models' performance with perturbed data

To assess the robustness and generalization capability of the trained models in Sect. 5.1 beyond the original training conditions, a comprehensive cross-model validation was performed. This evaluation tested the models' ability to maintain accurate predictions when the underlying physical parameters and sensor measurements deviated from the nominal training conditions. Five perturbation scenarios and a combined scenario incorporating all perturbation types simultaneously (excluding noise) were investigated as mentioned in Sect. 4.6. Each perturbation scenario was applied to three different data splits, and the previously trained models from ten weight initialization seeds were evaluated on each perturbed test set, resulting in 30 independent evaluations per model per scenario. Table 7 summarizes the performance metrics across all perturbation scenarios established in Sect. 4.6: 'Base' represents the nominal training results from Sect. 5.1; 'Restit.', 'Stiff.', and 'Hunt-C' correspond to the variations in restitution coefficient, contact stiffness, and contact model choice, respectively; 'All' denotes the consolidated set of physical perturbations; and 'Noise' refers to the addition of Gaussian sensor noise.

Comparing perturbed performance to the baseline results in Sect. 5.1, all models maintained relatively stable predictions under perturbations. LSTM and GRU demonstrated robust performance across all perturbation scenarios, with NMAE remaining close to baseline values (LSTM baseline: 0.49%; GRU baseline: 0.47%) and maintaining NMAE below 0.6% (95% CI: [0.45%, 0.56%] for LSTM; [0.42%, 0.56%] for GRU), even under combined perturbations. Both recurrent models showed comparable robustness across scenarios, with Median AE consistently below 4μm and 95th Percentile AE below 15μm, indicating that typical errors remain small. The restitution coefficient perturbation caused the largest performance degradation, with NRMSE increasing from baseline 0.8% to 1.48% for LSTM and from 0.74% to 1.44% for GRU. In contrast, stiffness variation and Gaussian noise perturbations had minimal impact, with performance nearly matching baseline levels.

Table 8 Performance metrics of models at joint B

Model	NMAE (%)	NRMSE (%)	Training time (s)	Eval. time (s)
TDNN	1.12 ± 0.14	1.62 ± 0.16	310.03 ± 80.12	0.19 ± 0.002
LSTM	0.38 ± 0.13	0.59 ± 0.17	1511.79 ± 320.45	0.90 ± 0.02
GRU	0.44 ± 0.12	0.64 ± 0.16	1380.01 ± 299.24	0.92 ± 0.09

In contrast, TDNN experienced substantially higher errors across all scenarios, with NMAE ranging from 1.42% to 1.64% and NRMSE ranging from 2.19% to 3.21%. The 95th Percentile AE for TDNN ranged from 40.68 to 47.14μm, showing that even the worst-case predictions have significantly larger deviations.

5.3 Evaluating model performance for different clearance locations

The performance of the methods was also evaluated when the clearance was located at joint B in Fig. 1. For this particular case, the same four accelerometer sensors as in Sect. 5.1 (ID 5, 6, 7, and 8) were used. The three models, TDNN, LSTM, and GRU, were assessed with the same 30-run cross-validation methodology. Table 8 represents performance statistics for each model at joint B.

Similar to Sect. 5.1, statistical comparisons were performed using a paired t-test with a significant level of $\alpha = 0.01$. LSTM achieved an NMAE of $0.38 \pm 0.13\%$ (95% CI: [0.33%, 0.43%]) and NRMSE of 0.59 ± 0.17 (95% CI: [0.52%, 0.65%]), while GRU obtained an NMAE of $0.44 \pm 0.12\%$ (95% CI: [0.39%, 0.48%]) and NRMSE of 0.64 ± 0.16 (95% CI: [0.58%, 0.67%]). LSTM and GRU demonstrated comparable performance, with no statistically significant difference for primary metrics (NMAE: $p = 0.043$; NRMSE: $p = 0.209$). Moreover, both LSTM and GRU significantly outperformed TDNN (mean NMAE: $1.12 \pm 0.14\%$, 95% CI: [1.07%, 1.18%]; mean NRMSE: $1.62 \pm 0.16\%$, 95% CI: [1.57%, 1.68%]) with $p < 0.001$ for all metrics.

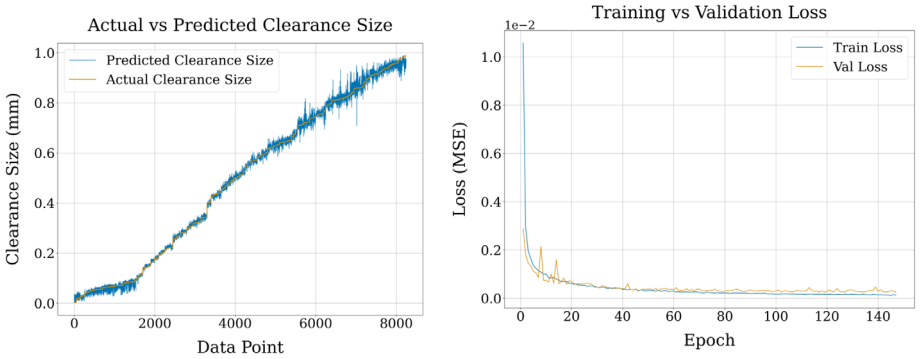
Comparing joint B results to the baseline (joint C), a notable performance was observed. At joint B, the LSTM model achieved a slightly lower NMAE (0.38%), compared to the GRU one (0.44%), whereas at joint C, GRU demonstrated a slightly better performance (NMAE: 0.47% vs LSTM: 0.49%). This location-dependent performance suggests that different joints exhibit distinct vibrational characteristics. Joint B's dynamics may favor the LSTM model's more complex gating mechanism for capturing long-term dependencies, while joint C's patterns are better suited to the GRU model's architecture. Additionally, all three models achieved substantially better performance at joint B compared to joint C.

Figure 9 shows representative predictions from a single run (data split seed 98, weight initialization seed 42) for each model at joint B.

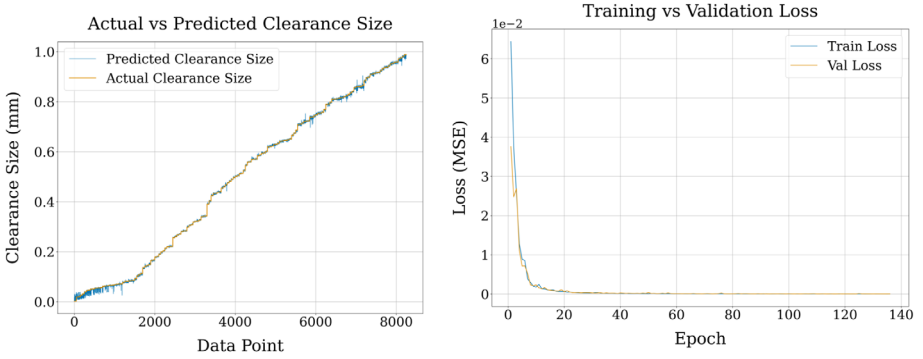
5.4 Evaluation of different sensor configurations

In industrial applications, it is important to provide solutions that are as unintrusive as possible and at a lower cost. Hence, it is of interest to find the minimal set of sensors required to predict accurate values of the clearance size.

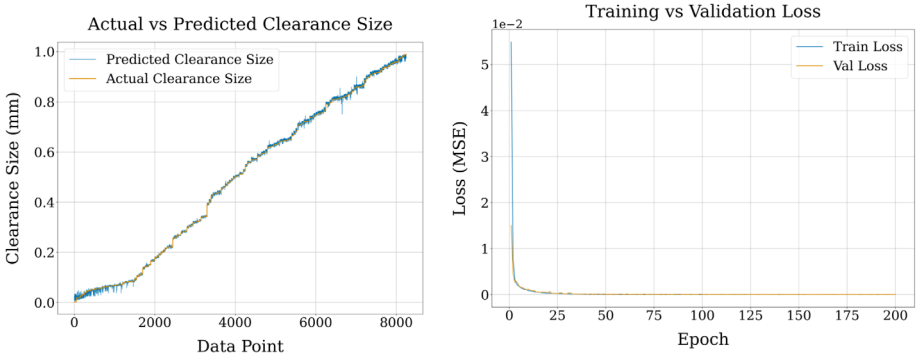
In this work, a total of 10 acceleration measurements are considered, offering information of acceleration in different positions and directions of the mechanism. In order to find a reduced set of sensors, a sensitivity analysis was performed based on the LSTM network, with the same hyperparameters and clearance location presented in Sect. 5.1. Starting with



(a) TDNN.



(b) LSTM.



(c) GRU.

Fig. 9 Performance comparison between true and predicted clearance sizes for neural network models at joint B using the 4-sensor configuration: (a) TDNN, (b) LSTM, and (c) GRU (Color figure online)

the full set of accelerometers, various tests were executed in which the number of measurements was reduced across five configurations: 10 sensors distributed across all components, 8 sensors (excluding slider), 6 sensors (excluding crank), 6 sensors (excluding connecting rod), and 4 sensors on crank only.

For the analysis, a limited evaluation with three runs (data split seed 42, 98, and 1024; weight initialization seed 42) was performed for each configuration due to computational

Table 9 Performance metrics for different sensor configurations at joint C

Configuration	Model	NMAE (%)	NRMSE (%)	Train (s)	Eval (s)
Baseline: 4 sensors (rod)	LSTM	0.39 ± 0.16	0.67 ± 0.11	1640.03 ± 363.26	0.89 ± 0.01
	GRU	0.59 ± 0.28	0.84 ± 0.28	1457.04 ± 522.17	0.88 ± 0.05
10 sensors (all)	LSTM	0.51 ± 0.15	0.77 ± 0.21	992.47 ± 294.44	0.81 ± 0.01
	GRU	0.45 ± 0.10	0.75 ± 0.02	900.62 ± 324.64	0.80 ± 0.001
8 sensors (no slider)	LSTM	0.45 ± 0.09	0.77 ± 0.07	1033.55 ± 267.07	0.79 ± 0.01
	GRU	0.40 ± 0.08	0.70 ± 0.11	1051.64 ± 314.02	0.78 ± 0.001
6 sensors (no crank)	LSTM	0.40 ± 0.07	0.64 ± 0.16	1257.46 ± 384.28	0.78 ± 0.003
	GRU	0.40 ± 0.10	0.67 ± 0.22	1254.95 ± 414.13	0.76 ± 0.01
6 sensors (no rod)	LSTM	0.42 ± 0.04	0.64 ± 0.07	1604.96 ± 522.87	0.87 ± 0.17
	GRU	0.46 ± 0.24	0.71 ± 0.33	1060.28 ± 365.53	0.77 ± 0.001
4 sensors (crank only)	LSTM	0.63 ± 0.21	1.30 ± 0.65	1337.83 ± 979.32	0.75 ± 0.01
	GRU	0.47 ± 0.11	0.87 ± 0.17	1410.01 ± 280.89	0.75 ± 0.01

constraints. For fair comparison, the baseline 4-sensor configuration (sensors on connecting rod only) was evaluated using the same three runs extracted from the comprehensive 30-run evaluation reported in Sect. 5.1. While this provides initial insights into sensor configuration effects, the reduced sample size results in higher uncertainty compared to the 30-run baseline statistics.

Table 9 presents the performance statistics for each sensor configuration. The baseline 4-sensor configuration achieved NMAE of $0.39 \pm 0.36\%$ for LSTM and $0.59 \pm 0.28\%$ for GRU across the three selected runs. Additionally, the GRU baseline shows higher variability due to one challenging data split (seed 1024: 0.91% NMAE), while the other two splits achieved typical performance (seed 42 and 98: 0.41-0.45% NMAE). In contrast, LSTM demonstrated more consistent performance across all three data splits.

The results, summarized in Table 9, show that the accuracy of clearance size estimation is dependent on sensors installed near the clearance location. For both the LSTM and GRU models, the 6-sensor configuration excluding crank sensors achieved the best performance (NMAE: $0.40 \pm 0.07\%$ for LSTM; $0.40 \pm 0.10\%$ for GRU), performing comparably to the baseline 4-sensor configuration. The 8-sensor configuration (excluding slider) also showed strong performance, with GRU achieving NMAE of $0.40 \pm 0.08\%$. The worst-case scenario occurred when sensors were only installed on the crank (NMAE: $0.63 \pm 0.21\%$ for LSTM; $0.47 \pm 0.11\%$ for GRU), which is consistent with the fact that these measurements are far from the joint affected by the clearance. Interestingly, the 10-sensor configuration using all available measurements did not yield the best performance, with LSTM achieving NMAE of $0.51 \pm 0.15\%$ and GRU achieving $0.45 \pm 0.10\%$. This suggests that additional sensors distant from the clearance location may introduce noise rather than useful information.

When the accelerations in the connecting rod or slider are measured, almost any configuration yields accurate estimations. However, given the limited sample size, these should be interpreted cautiously, and further validation with additional runs would be needed.

6 Conclusions

Joint clearance is unavoidable in machines and mechanisms due to manufacturing tolerances, wear, and assembly imperfections, and it can strongly affect dynamic response and fault-critical behavior. This work investigated the data-driven estimation of clearance size in a slider–crank mechanism using simulated acceleration measurements from a small number of accelerometers placed at selected locations.

We compared classical machine learning models (RF, XGBoost) with deep neural networks (TDNN, GRU, and LSTM). The results show that deep neural networks substantially improve clearance size estimation compared to classical models. In the four-sensor configuration with accelerometers mounted on the connecting rod, the normalized mean absolute error decreased from 9.3% for XGBoost to $0.49 \pm 0.14\%$ for the LSTM model and $0.47 \pm 0.18\%$ for GRU across 30 independent runs, while the evaluation time increased from 0.01 s to approximately 0.9 s for both models. This indicates that the additional computational cost of deep learning remains acceptable for diagnostic applications, while yielding an order-of-magnitude gain in accuracy. Moreover, among deep architectures evaluated at the baseline clearance location (joint C), LSTM and GRU demonstrated comparable performance with no statistically significant difference in primary metrics (paired t-test, $p > 0.01$), both substantially outperforming TDNN (NMAE: $1.43 \pm 0.19\%$; NRMSE: $2.19 \pm 0.27\%$; $p < 0.001$). However, TDNN offered a computational advantage with approximately 81% reduction in training time, though at the cost of lower accuracy. Cross-model validation demonstrated robust performance across various perturbation scenarios, with both LSTM and GRU maintaining NMAE below 0.6%, confirming the models' generalization capability beyond nominal training conditions. Preliminary exploration of different sensor configurations, which was evaluated with limited experiments, suggested that strategic sensor placement is more critical than sensor quantity. However, these results are based on limited runs and require further investigation with larger sample sizes to establish statistical significance and confirm optimal sensor placement strategies.

The main limitation of this study is that the models were trained and validated solely on simulated data for a single mechanism and a specific clearance location. Although realistic modeling assumptions and sensor characteristics were employed, experimental validation on a physical test rig is essential to assess robustness to measurement noise, modeling uncertainties, and varying operating conditions. In addition, validation against experimental data will involve a model characterization, which may include the modeling of spatial clearances instead of planar ones. Future work will therefore focus on experimental studies and integration of the proposed models into online monitoring and digital-twin frameworks for fault diagnostics.

In summary, the study demonstrates that deep learning models, and in particular LSTM and GRU networks, can provide accurate and practically feasible clearance size detection from acceleration measurements, offering a strong alternative to traditional analytical and classical machine learning approaches in joint fault diagnostics.

Acknowledgements The authors acknowledge the financial support of the Finnish Ministry of Education and Culture through the Intelligent Work Machines Doctoral Education Pilot Program (IWM VN/3137/2024-OKM-4), the support of project PID2022-139832NB-I00 (funded by MICIU/AEI/10.13039/501100011033 and ERDF, EU), and grant ED431C 2023/01 from the Government of Galicia.

Author contributions V.N: Model development, Experiments, Visualization; A.R: Multibody system model set up, Data curation and preparation; F.G: Supervision, Conceptualization; A.M: Supervision, Conceptualization; J.K: Conceptualization; G.O: Supervision, Conceptualization, Technical guidance; All authors contributed to writing, review, editing, and approved the final manuscript.

Funding information Open Access funding provided by LUT University (previously Lappeenranta University of Technology (LUT)).

Data availability The simulation data used in this study were generated using a multibody dynamics model and are available from the corresponding author upon reasonable request.

Declarations

Competing interests The authors declare no competing interests.

References

1. Tian, Q., Flores, P., Lankarani, H.M.: A comprehensive survey of the analytical, numerical and experimental methodologies for dynamics of multibody mechanical systems with clearance or imperfect joints. Mech. Mach. Theory **122**, 1–57 (2018). https://doi.org/10.1016/j.mechmachtheory.2017.12.002
2. Flores, P., Koshy, C.S., Lankarani, H.M., Ambrósio, J., Claro, J.C.P.: Numerical and experimental investigation on multibody systems with revolute clearance joints. Nonlinear Dyn. **65**(4), 383–398 (2011). https://doi.org/10.1007/s11071-010-9899-8
3. Erkaya, S., Doğan, S., Şefkatlıoğlu, E.: Analysis of the joint clearance effects on a compliant spatial mechanism. Mech. Mach. Theory **104**, 255–273 (2016). https://doi.org/10.1016/j.mechmachtheory.2016.06.009
4. Wuweikai, X., Shaoze, Y., Jianing, W., Wendong, N.: Dynamic response and sensitivity analysis for mechanical systems with clearance joints and parameter uncertainties using Chebyshev polynomials method. Mech. Syst. Signal Process. **138**, 106596 (2020). https://doi.org/10.1016/j.ymssp.2019.106596
5. Wuweikai, X., Shaoze, Y., Jianing, W.: A comprehensive method for joint wear prediction in planar mechanical systems with clearances considering complex contact conditions. Sci. China, Technol. Sci. **58**(1), 86–96 (2015). https://doi.org/10.1007/s11431-014-5685-z
6. Xu, L., Han, Y., Dong, Q., Jia, H.: An approach for modelling a clearance revolute joint with a constantly updating wear profile in a multibody system: simulation and experiment. Multibody Syst. Dyn. **45**(4), 457–478 (2019). https://doi.org/10.1007/s11044-018-09655-z
7. Mobley, R.K., Higgins, L.R., Wikoff, D.J.: Maintenance Engineering Handbook, 7th edn. McGraw Hill, New York (2008)
8. Wumaier, T., Xu, C., Guo, H., Jin, Z., Zhou, H.: Fault diagnosis of wind turbines based on a support vector machine optimized by the sparrow search algorithm. IEEE Access **9**, 69307–69315 (2021). https://doi.org/10.1109/ACCESS.2021.3075547
9. Balaji, P.A., Sugumaran, V.: Fault detection of automobile suspension system using decision tree algorithms: a machine learning approach. Proc. Inst. Mech. Eng., E J. Process Mech. Eng. **238**, 1206–1217 (2023). https://doi.org/10.1177/09544089231152698
10. Ahmed Faris, A., Oudira, H., Aissa, C., Kichou, S.: Faults detection and diagnosis of PV systems based on machine learning approach using random forest classifier. Energy Convers. Manag. **118076** (2024). https://doi.org/10.1016/j.enconman.2024.118076
11. Verma, A., Nagpal, S., Desai, A., Sudha, R.: An efficient neural-network model for real-time fault detection in industrial machine. Neural Comput. Appl. **33**, 1297–1310 (2021). https://doi.org/10.1007/s00521-020-05033-z
12. Zhuang, L., Xu, A., Wang, X.-L.: A prognostic driven predictive maintenance framework based on Bayesian deep learning. Reliab. Eng. Syst. Saf. **234**, 109181 (2023). https://doi.org/10.1016/j.ress.2023.109181

13. Ayvaz, S., Alpay, K.: Predictive maintenance system for production lines in manufacturing: a machine learning approach using IoT data in real-time. Expert Syst. Appl. **173**, 114598 (2021). https://doi.org/10.1016/j.eswa.2021.114598
14. Choi, H.-S., An, J., Kim, J.-G., Jung, J.-Y., Choi, J., Orzechowski, G., Mikkola, A., Choi, J.H.: Data-driven simulation for general purpose multibody dynamics using deep neural networks. Multibody Syst. Dyn. **51**, 419–454 (2021). https://doi.org/10.1007/s11044-020-09772-8
15. Pan, Y., Nie, X., Li, Z., Gu, S.: Data-driven vehicle modeling of longitudinal dynamics based on a multibody model and deep neural networks. Measurement **180**, 109541 (2021). https://doi.org/10.1016/j.measurement.2021.109541
16. Nasr, A., Bell, S., He, J., Whittaker, R., Jiang, N., Dickerson, C., McPhee, J.: MuscleNET: mapping electromyography to kinematic and dynamic biomechanical variables by machine learning. J. Neural Eng. **18**, 0460 (2021). https://doi.org/10.1088/1741-2552/ac1adc
17. Han, S., Orzechowski, G., Kim, J.-G., Mikkola, A.: Data-driven friction force prediction model for hydraulic actuators using deep neural networks. Mech. Mach. Theory **192**, 105545 (2024). https://doi.org/10.2139/ssrn.4494358
18. Chin, K., Hellebrekers, T., Majidi, C.: Machine learning for soft robotic sensing and control. Adv. Intell. Syst. **2** (2020). https://doi.org/10.1002/aisy.201900171
19. Erkaya, S., Uzmay, I.: A neural–genetic (NN–GA) approach for optimising mechanisms having joints with clearance. Multibody Syst. Dyn. **20**(1), 69–83 (2008). https://doi.org/10.1007/s11044-008-9106-6
20. Bauchau, O.A., Rodriguez, J.: Modeling of joints with clearance in flexible multibody systems. Int. J. Solids Struct. **9**, 41–63 (2002). https://doi.org/10.1016/S0020-7683(01)00186-X
21. Ma, J., Qian, L.: Modeling and simulation of planar multibody systems considering multiple revolute clearance joints. Nonlinear Dyn. **90**, 1907–1940 (2017). https://doi.org/10.1007/s11071-017-3771-z
22. Wang, J., Zhou, S., Wu, J., Qing, J., Kang, T., Shao, J.: Dynamic modeling and analysis of flexible-joint robots with clearance. Sensors **24** (2024)
23. Cheng, S.: Dynamic analysis of reciprocating compressor with revolution joint clearance and translational joint clearance. In: 2021 Global Reliability and Prognostics and Health Management (PHM-Nanjing), pp. 1–6 (2021). https://doi.org/10.1109/PHM-Nanjing52125.2021.9612815
24. Tan, H., Li, L., Huang, Q., Jiang, Z., Li, Q., Zhang, Y., Yu, D.: Influence of two kinds of clearance joints on the dynamics of planar mechanical system based on a modified contact force model. Sci. Rep. **13** (2023). https://doi.org/10.1038/s41598-023-47315-1
25. Bai, Z., Ning, Z., Zhou, J.: Study on wear characteristics of revolute clearance joints in mechanical systems. Micromachines **13** (2022). https://doi.org/10.3390/mi13071018
26. Jia, Y., Chen, X.: Dynamic response analysis for multi-degrees-of-freedom parallel mechanisms with various types of three-dimensional clearance joints. Int. J. Adv. Robot. Syst. **18** (2021). https://doi.org/10.1177/17298814211017716
27. Bauchau, O.A.: Flexible Multibody Dynamics. Springer, Dordrecht (2011). https://doi.org/10.1007/978-94-007-0335-3
28. Flores, P., Ambrósio, J., Lankarani, H.M.: Contact-impact events with friction in multibody dynamics: back to basics. Mech. Mach. Theory **184**, 105305 (2023). https://doi.org/10.1016/j.mechmachtheory.2023.105305
29. Chaturvedi, E., Sandu, C., Sandu, A.: A nonsmooth dynamics framework for simulating frictionless spatial joints with clearances. Multibody Syst. Dyn. **63**(1–2), 3–37 (2024). https://doi.org/10.1007/s11044-024-09971-7
30. Hunt, K.H., Crossley, F.R.E.: Coefficient of restitution interpreted as damping in vibroimpact. J. Appl. Mech. **42**(2), 440–445 (1975). https://doi.org/10.1115/1.3423596
31. Lankarani, H.M., Nikravesh, P.E.: A contact force model with hysteresis damping for impact analysis of multibody systems. J. Mech. Des. **112**(3), 369–376 (1990). https://doi.org/10.1115/1.2912617
32. González, F., Kövecses, J., Font-Llagunes, J.M.: Load assessment and analysis of impacts in multibody systems. Multibody Syst. Dyn. **38**(1), 1–19 (2015). https://doi.org/10.1007/s11044-015-9485-4
33. Flores, P., Ambrósio, J.: Revolute joints with clearance in multibody systems. Comput. Struct. **82**(17), 1359–1369 (2004). https://doi.org/10.1016/j.compstruc.2004.03.031
34. García de Jalón, J., Bayo, E.: Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge. Springer, New York (1994). https://doi.org/10.1007/978-1-4612-2600-0
35. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
36. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD'16, pp. 785–794. Association for Computing Machinery, USA (2016). https://doi.org/10.1145/2939672.2939785
37. Bebis, G., Georgiopoulos, M.: Feed-forward neural networks. IEEE Potentials **13**(4), 27–31 (1994). https://doi.org/10.1109/45.329294

38. Waibel, A.: Modular construction of time-delay neural networks for speech recognition. Neural Comput. **1**(1), 39–46 (1989). https://doi.org/10.1162/neco.1989.1.1.39
39. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
40. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
41. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **6**, 107–116 (1998). https://doi.org/10.1142/S0218488598000094
42. Cho, K., Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder–decoder approaches. In: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pp. 103–111. Association for Computational Linguistics, Qatar (2014)
43. Analog Devices: Data Sheet ADXL 1001/ADXL 1002: Low Noise, High Frequency MEMS Accelerometers. https://www.analog.com/media/en/technical-documentation/data-sheets/adxl1001-1002.pdf Accessed 2025-12-11
44. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: ICLR - International Conference on Learning Representations, San Diego, USA (2015)
45. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 15, pp. 315–323. PMLR, USA (2011)