Structural and Multidisciplinary Optimization (2023) 66:96
https://doi.org/10.1007/s00158-023-03553-5

RESEARCH PAPER

# Machine learning enabled fusion of CAE data and test data for vehicle crashworthiness performance evaluation by analysis

Jice Zeng[1] · Guosong Li[2] · Zhenyan Gao[2] · Yang Li[2] · Srinivasan Sundararajan[2] · Saeed Barbat[2] · Zhen Hu[1]

Received: 28 November 2022 / Revised: 19 February 2023 / Accepted: 13 March 2023 / Published online: 8 April 2023


## Abstract

Evaluation and potential certification by analysis (CBA) has emerged as a promising tool to reduce costly crashworthiness tests of prototype, by using physics-based simulation and analysis. One of the major obstacles in the CBA implementation for vehicle crashworthiness certification is the discrepancy between computer-aided engineering (CAE) model prediction and the actual response of vehicle crash tests. This paper aims to improve the prediction accuracy of the crashworthiness CAE model under new test speed by fusing CAE data and a small number of crash test data using machine learning models. Two novel approaches are developed in the time domain and the displacement domain to achieve the goal of data fusion for the correction of the CAE model. More specifically, in the time domain (time vs. deceleration), a low-fidelity temporal convolutional network (TCN) is first trained using a large volume of simulation data generated from a CAE model. The low-fidelity TCN model is then fine-tuned to a multi-fidelity TCN through transfer learning using a small number of test data. In the displacement domain (displacement vs. deceleration), the deceleration response as a function of time is first converted into a function of displacement. A Gaussian process regression (GPR) model is then utilized to capture the model bias of the nonlinear spring constant under a dynamic analysis scheme. The learned GPR model finally is integrated into vehicle deceleration dynamic analysis along with the original CAE model to predict vehicle deceleration under a new crash speed. A real-world case study of vehicle crash tests is employed to validate the proposed two approaches. The results show that both proposed approaches in general can recover model bias of the CAE model and give a noticeable improvement in CAE model prediction accuracy.

**Keywords** CAE · Model bias · Model updating · Temporal convolutional network · Gaussian process

## 1 Introduction

A new vehicle design is usually evaluated under various loadings conditions concerning vehicle crashworthiness, energy efficiency, durability, style flexibility, safety, etc. A successful vehicle design must satisfy a variety of requirements and criteria, such as safety regulations, structural performance criteria, economic cost, and environmental standards (Happian-Smith 2001). Among all evaluation metrics, vehicle crashworthiness receives considerable attention and is an essential design attribute in the automotive industry to guarantee the integrity of vehicle structure, the minimum cost in vehicle design, and more importantly occupants' safety in the event of a crash (Fang et al. 2017). In general, vehicle crashworthiness represents the structural ability to manage crash energy through controlled deformation, and thereby protecting occupants during an impact by reducing the risk of fatalities or serious injuries with optimized structures and restraint systems.

Crashworthiness design is a challenging task as it involves a tradeoff between various vehicular attributes, an example of that is vehicle light weighting and vehicle crashworthiness (Sinha et al. 2007). Motivated by ensuring vehicle crashworthiness, tremendous efforts have been made by researchers in academia and major automobile original

Responsible Editor: Hongyi Xu

✉ Zhen Hu
zhennhu@umich.edu

[1] Department of Industrial and Manufacturing Systems Engineering, University of Michigan-Dearborn, 2340 HPEC, Dearborn, MI 48128, USA

[2] Vehicle Structure and Safety Research Department, Research & Advanced Engineering, Ford Motor Company, Dearborn, MI 48126, USA

equipment manufacturers (OEMs) in both the early-design stage and the production stage. In the *early-design* stage, various approaches have been developed in the past decades to optimize the structural design to improve vehicle crashworthiness. In this context, vehicle structural design for crashworthiness improvement is usually formulated as an optimization problem with: (1) various performance metrics, e.g., deformation, acceleration, and the probability of injury risk, as design objectives, and (2) geometric dimensions in structural members (e.g., frontal-end including, the bumper, energy absorbing box, hood, and fender) as design variables. For example, Sun et al. (2017) proposed a crisscross thin-walled tube structure with superior crash energy absorption and employed the non-dominated sorting genetic algorithm II (NSGA-II) to optimize the crisscross shape. Yin et al. (2011) adopted a multi-objective particle swarm optimization algorithm to maximize the capacity of energy absorption for honeycomb-filled tubes and minimize the peak crushing force. Similarly, Lanzi et al. (2004) presented an optimization framework for the shape of conical absorbers under different loading conditions using genetic algorithms along with response surfaces. Li et al. (2021) developed a multi-objective evolutionary algorithm coupled with a self-organizing decomposition selection strategy for the crashworthiness design. In addition, Gu et al. (2013) conducted a comparative study on different multi-objective optimization methods in the case of frontal crash scenarios and shed light on reliability and robustness issues of vehicle crashworthiness design. In the *production* stage, efforts mainly concentrate on achieving crashworthiness requirements using prototype vehicle tests to ensure that a new vehicle design can meet those requirements in reality. Vehicle crashworthiness certification requires many prototype vehicle tests, which are costly. With the development of high-fidelity computer simulation models, computational mechanics, and multi-body dynamics simulation in recent years, certification by analysis (CBA) has emerged as a promising tool to reduce the required number of costly prototype vehicle tests in the certification process. CBA is a process of supporting the certification of structural components by virtual physics-based simulations (e.g., computer-aided engineering (CAE) simulations) (Guida et al. 2018). The concept of CBA was originally proposed in the aerospace sector and has largely applied to the certification of aircraft structural components (Caputo et al. 2021; Jackson et al. 2006; Olivares et al. 2010). Similar practices have also been adopted in the automobile industry. It has great potential to efficiently deploy and use prototype resources for developing new vehicles in automobile companies.

A high-fidelity CAE simulation of vehicle crashworthiness is an essential element for both the crashworthiness improvement in the early-design stage and/or the crashworthiness certification using CBA in the production stage. A long-standing research question that needs to be addressed is the imperfection of the CAE models, due to missing physics, numerical approximations, assumptions and simplifications in modeling, and the highly complicated vehicle crash process. The discrepancy of the CAE crashworthiness simulation model significantly affects not only the effectiveness of structural design optimization in the early design, but also the adoption of CBA in the post design for reducing the number of costly prototype vehicle tests. Inaccurate prediction of the CAE model could lead to not well designed vehicle structure and a prolonged vehicle development process (Gu et al. 2013). Aiming to make the prediction of CAE crashworthiness simulation model closer to the response of actual vehicle crash tests, a few approaches have been developed in recent years. Zhan et al. (2014) proposed a Bayesian inference-based interpolation and extrapolation method combined with response surface model for a vehicle crash design. Xi et al. (2014) developed a copula-based approach for model bias correction of vehicle crash simulation by establishing a statistical relation between CAE model bias and outputs of a full-frontal crash scenario of simulation. Wang and Shi (2014) introduced a bias function for vehicle crashworthiness design by using a Gaussian process regression (GPR) model to capture the difference between observations and model-derived outputs. Furthermore, Zhan et al. (2013) investigated and compared different stochastic methods for model bias correction with an application to vehicle impact tests.

In addition, machine learning (ML) model-based approaches have emerged as promising tools to address the challenges arising in vehicle crashworthiness design, such as model nonlinearity and implicit model form. ML also enables to assist various aspects in developing CAE model in terms of more properly modeling materials and accelerating CAE model computation. Kohar et al. (2021) used 3D convolutional neural networks (CNN) autoencoders together with long-short term memory to predict force–displacement responses for dynamic axial crushing of rectangular tubes with the application of vehicle crashworthiness. Tang et al. (2017) proposed parallel random forest ML approach to predict force–displacement curve given different train crash conditions in multi-body dynamics simulation. Du et al. (2021) also developed a new data-mining method for vehicle safety design, which efficiently realizes the vehicle design at multi-levels and build decision trees using big datasets of vehicle crash accounting for uncertainty involving in the design variables.

Despite partial success in the methods, there are still some limitations that need to be addressed. For example, sufficient test data are usually required in the current methods to ensure the effectiveness of model correction of CAE crash models. The acquisition of a large amount of crashworthiness data, however, may be very challenging since building

prototype vehicles for crash tests is economically very expensive. In addition, a lot of historical crash test data are collected in not well controlled test conditions, which leads to the crash tests poorly correlated or even uncorrelated with the CAE model. Furthermore, many current methods are restricted to model bias correction of CAE surrogate models using high-fidelity CAE simulations, instead of actual tests.

The goal of this paper is to overcome the limitations of current methods in correlating costly vehicle crash tests with high-fidelity CAE simulations to facilitate model-based structural design for crashworthiness improvement, and to enable for the reduction of crash tests through CBA. Two approaches are proposed to achieve this goal in time domain (time vs. deceleration) and displacement domain (displacement vs. deceleration). In the time domain, a low-fidelity temporal convolutional network (TCN) is first trained using a large volume of simulated data generated from a CAE model. The low-fidelity TCN model is then fine-tuned using a limited number of crash test data, resulting in a multi-fidelity TCN model for response prediction of vehicle crashes under different new speeds. The fusion of CAE data with test data is accomplished by directly minimizing the misfit between prediction from multi-fidelity TCN and test data, and therefore the approach in the time domain does not require solving any nontrivial differential equations. In the displacement domain, vehicle deceleration responses under different crash speeds are first converted into a function of displacement. The discrepancy between the predicted nonlinear spring constant and the actual spring constant is then analyzed based on the displacement domain response. A GPR model is then constructed to capture the unmodeled physics of the nonlinear spring constant. In the prediction phase, the constructed GPR model is integrated with the CAE model in a dynamic scheme to iteratively predict vehicle deceleration response under a new crash speed. A fundamental difference between the proposed two approaches is the treatment of time. In the time-domain method, time is treated as an input variable directly and the model becomes to be a static modeling problem. In the displacement-domain method, the vehicle crashworthiness analysis is solved as a dynamic analysis problem that involves solving differential equations in an iterative manner over time. The proposed approaches are validated using actual vehicle crash data and the results demonstrate the effectiveness of the proposed two approaches. The contributions of this paper can be summarized as: (1) a multi-fidelity TCN approach for the correlation of CAE crashworthiness data with actual test data; (2) a novel approach for correction of CAE crash simulation in the displacement domain; and (3) demonstration of the proposed method using a real-world case study.

The remainder of this paper is organized as follows. Section 2 briefly reviews the background of bias correction of the CAE crash simulation model. Section 3 presents the proposed two approaches in the time domain and displacement domain for the fusion of CAE data and test data. A real-world example of a vehicle crash test is used to demonstrate the effectiveness of two approaches in Sect. 4. Finally, the conclusion and future work are presented in Sect. 5.

## 2 Background

As discussed in Sect. 1, this paper is motivated by correlating the CAE crash simulation model with actual crash test to make the CAE model closer to the actual physics. Figure 1 shows the physical test and a CAE model for the full-frontal impact of a vehicle. A successful vehicle design

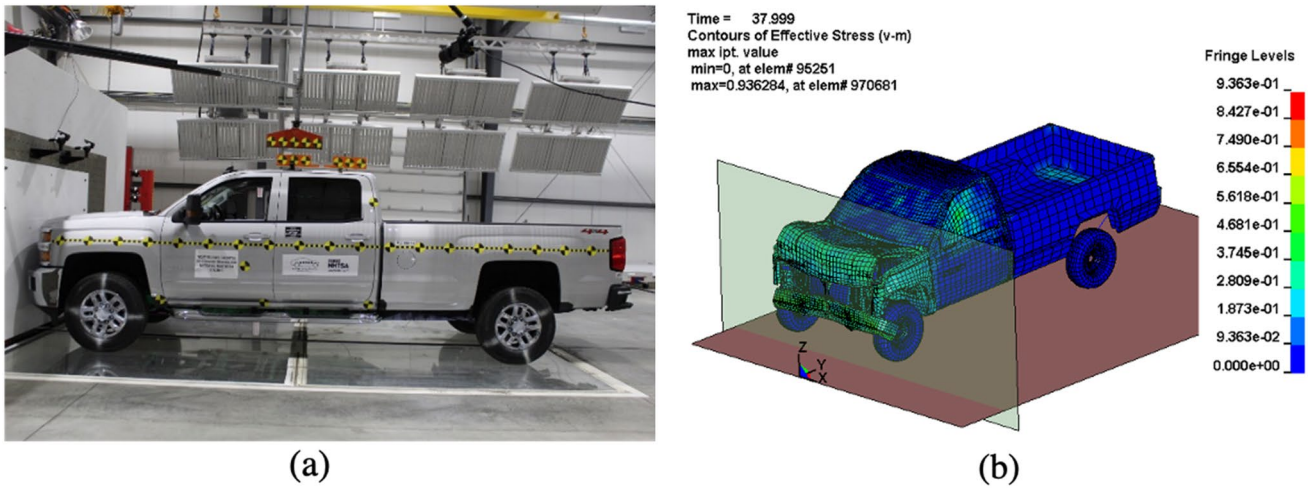


**Fig. 1** Vehicle crashworthiness design: **a** physical test, and **b** CAE model

must maximally attenuate front impact to an allowable level by absorbing impact energy transmitted to the vehicle to minimize the vehicle damage to occupant compartments and ensure occupants' safety.

Due to various reasons, the CAE model in Fig. 1b may not accurately represent the actual test in Fig. 1a. Some potential reasons include: (1) a CAE is usually built based on available knowledge of the physical systems and may involve subjective assumptions and simplifications. The credibility of a CAE model largely depends on the degree of knowing the reality by engineers. (2) Model parameters in the current model may fit well for past tests but mis-capture the information from the new test. The mismatch between the CAE model and the actual test is usually referred to as model bias, model discrepancy, model uncertainty, and model form error (Jiang et al. 2020).

Attributed to the importance of improving the prediction accuracy of CAE model in both design optimization and CBA, a wealth of literature discusses the underlying causes of model uncertainty and how model uncertainty can be mathematically characterized (Arendt et al. 2012; Zeng and Kim 2020, 2022; Zeng et al. 2022; Thelen et al. 2023). Amongst all current methods, the work by Kennedy and O'Hagan (2001) is one of the methods worth highlighting. They comprehensively summarized different sources of model uncertainty into several categories including (1) unknown parameters in a CAE model, such as geometric and material properties in structural elements; (2) model bias that is induced by the missing physics; (3) interpolation error due to insufficient tests that are usually economically expensive and CAE simulations with high computing resources; (4) numerical approximations in a CAE model, e.g., numerical integration; (5) variability of experimental test, which is inevitably embodied in any tests and mainly affected by human error and equipment precision. A widely used mathematical model for correlating the CAE model with test data suggested by Kennedy and O'Hagan is given as follows (Kennedy and O'Hagan 2001; Thelen et al. 2022)

$$y_o(\mathbf{x}) = \rho g(\mathbf{x}, \boldsymbol{\theta}) + \delta(\mathbf{x}) + \varepsilon, \tag{1}$$

where $y_o(\mathbf{x})$ is experimental observation given a set of inputs $\mathbf{x}$, $y = g(\mathbf{x}, \boldsymbol{\theta})$ is CAE model with a set of inputs $\mathbf{x}$ and model parameters $\boldsymbol{\theta}$, $\delta(\mathbf{x})$ is model bias/model discrepancy, $\rho$ is an unknown regression coefficient, and $\varepsilon$ is measurement error which is usually assumed as Gaussian noise and independent from $\mathbf{x}$ and $\boldsymbol{\theta}$. The difference between $\mathbf{x}$ and $\boldsymbol{\theta}$ is that $\mathbf{x}$ can be those controllable variables that execute different operation configurations, e.g., excitation or size/shape of a product and $\boldsymbol{\theta}$ is a vector of model parameters that are random and uncontrollable, such as material properties, structural mass, and stiffness. The formulation given in Eq. (1) thoroughly takes all possible uncertainty sources into consideration. With this formulation, the target of model uncertainty quantification is split into model bias correction which can be performed by estimating the term of $\delta(\mathbf{x})$ and calibration of uncertain model parameters $\boldsymbol{\theta}$.

Vehicle crash tests data in this paper are obtained from historical tests. Very limited information is available in the record. The selection of uncertain model parameters, therefore, becomes ambiguous and challenging. Thus, we only focus on model bias correction without considering calibration of $\boldsymbol{\theta}$ in this paper. The objective becomes estimating $\delta(\mathbf{x})$ to improve the accuracy of the CAE model. Then, the formulation in Eq. (1) is simplified as

$$y_o(\mathbf{x}) = g(\mathbf{x}) + \delta(\mathbf{x}) + \varepsilon. \tag{2}$$

Even though the formulation of model correction given in Eq. (2) looks very straightforward, the actual implementation for vehicle crashworthiness CAE model correction is not as simple as shown in this equation. It should be noted that the unknown regression factor $\rho$ in Eq. (1) could be estimated simultaneously with the unknown hyperparameters of the model discrepancy terms or could be simply set as $\rho = 1$ to simplify the implementation. When $\rho$ is set as one, Eq. (2) can be considered a special case of Eq. (1) and is very widely used in the literature (Jiang et al. 2020). In this paper, the special case $\rho = 1$ is employed to simplify the training procedure of the ML models for the discrepancy term. In Sect. 3, we discuss how the proposed two approaches correlate CAE data and actual crash test data in the time domain and displacement domain, respectively.

## 3 Proposed approaches

The objective of the proposed approaches is to fuse data from CAE model with limited test data to bring CAE model closer to the actual physics. Figure 2 shows the overall workflow of the proposed approaches which consists of two main steps, namely ML modeling and applying ML models.

In the ML modeling step, a large volume of data is first generated from a CAE simulation model. Then a small amount of test data is used in conjunction with the CAE data to train ML models. Different ways of fusing CAE with test data for time domain and displacement domain approaches are proposed. In the time domain, the low-fidelity model is first built only using CAE data, then fine-tuned into a multi-fidelity model using test data. In the displacement domain, CAE data in companion with test data are used to formulate the model bias function. Details on data fusion are introduced in Sects. 3.1 and 3.2. After the training and validation of the ML models, the models are employed to predict response of vehicle structure at the event of crash under a new speed.

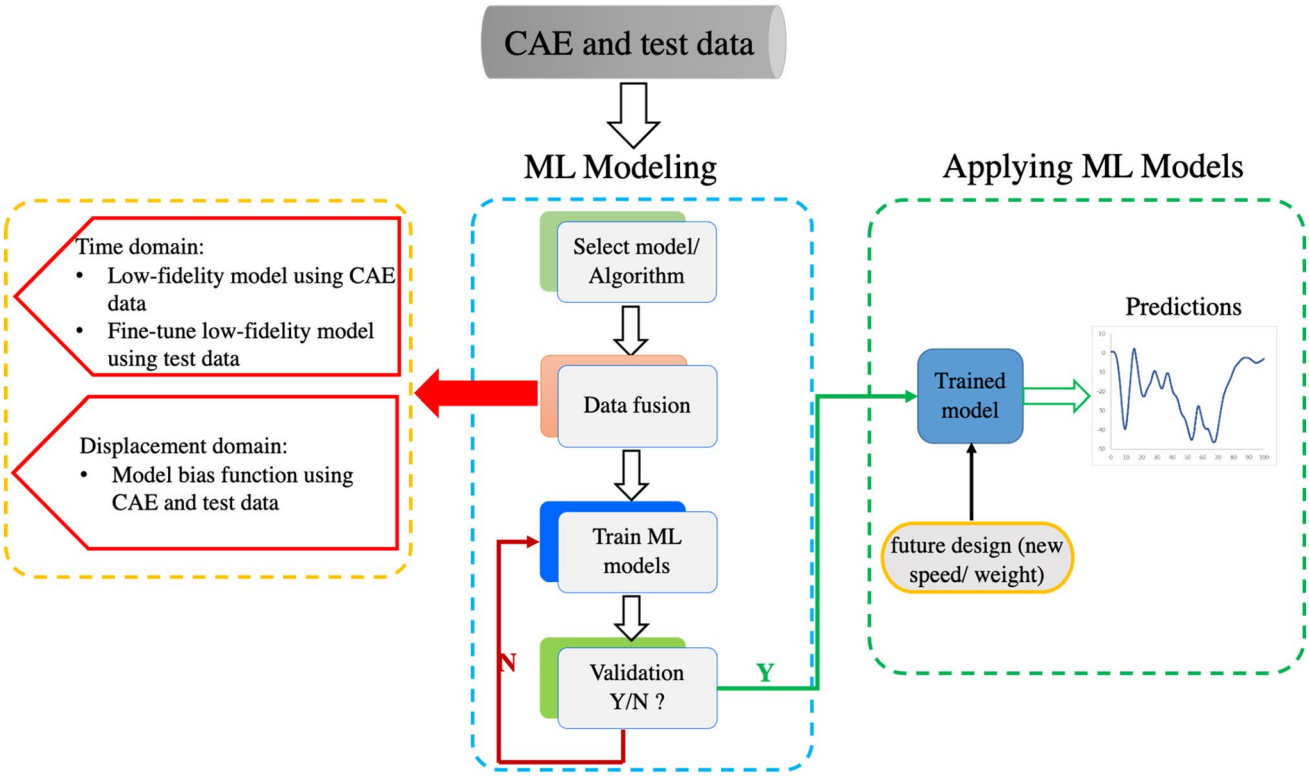


**Fig. 2** The workflow of ML training and prediction for data fusion

Before we discuss details for the proposed two approaches in the time domain and displacement domain, we define the models and variables used in this paper for the development. The quantity of interest in this paper is the deceleration of a vehicle at the event of crash and is denoted as $a_{\mathrm{d}}(t)$. In a discrete-time form, the deceleration at time instant $t_i$ is represented as $a_{d,i},\ i = 1, \ldots, N_t$, where $N_t$ is the total number of time steps in the analysis. As mentioned in Sect. 2, the input variables $\mathbf{x} = [v_0,\ w_{\mathrm{f}},\ w_{\mathrm{r}}] \in \mathbb{R}^3$ include only initial crash speed $v_0$, vehicle front weight and rear weight $(w_{\mathrm{f}},\ w_{\mathrm{r}})$, due to the lack of information from crash tests. Based on these definitions, we have the CAE crashworthiness analysis model as

$$\mathbf{a}_{\mathrm{s}} = G_{\mathrm{s}}(\mathbf{x}) = G_{\mathrm{s}}(v_0,\ w_{\mathrm{f}},\ w_{\mathrm{r}}), \tag{3}$$

where $G_{\mathrm{s}}(\mathbf{x})$ stands for the CAE model with inputs $\mathbf{x}$, and $\mathbf{a}_{\mathrm{s}} = [a_{\mathrm{s},1},\ a_{\mathrm{s},2}, \ldots, a_{\mathrm{s},N_t}] \in \mathbb{R}^{N_t}$ are the predicted deceleration response from one CAE analysis.

Assume that we generated $N_{\mathrm{e}}$ groups of CAE simulation data and collected $N_{\mathrm{c}}$ groups of deceleration data from actual crash tests ($N_{\mathrm{c}} \ll N_{\mathrm{e}}$), we then have the CAE data and test data as

- **CAE data**: $\mathbf{a}_{\mathrm{s}}^{(j)},\ j = 1, \ldots, N_{\mathrm{e}}$, where $\mathbf{a}_{\mathrm{s}}^{(j)} = [a_{\mathrm{s},1}^{(j)},\ a_{\mathrm{s},2}^{(j)}, \ldots, a_{\mathrm{s},N_t}^{(j)}] = G_{\mathrm{s}}(v_0^{(j)},\ w_{\mathrm{f}}^{(j)},\ w_{\mathrm{r}}^{(j)})$ are the deceleration data collected with the $j$th training sample $\mathbf{x}^{(j)} = [v_0^{(j)},\ w_{\mathrm{f}}^{(j)},\ w_{\mathrm{r}}^{(j)}],\ \forall j = 1, \ldots, N_{\mathrm{e}}$ of the input variables.
- **Test data**: $\mathbf{a}_{\mathrm{d}}^{(k)},\ k = 1, \ldots, N_{\mathrm{c}}$, where $\mathbf{a}_{\mathrm{d}}^{(k)} = [a_{\mathrm{d},1}^{(k)},\ a_{\mathrm{d},2}^{(k)}, \ldots, a_{\mathrm{d},N_t}^{(k)}]$ are the deceleration data collected with the $k$th crash test under testing condition $\mathbf{x}^{(k)} = [v_0^{(k)},\ w_{\mathrm{f}}^{(k)},\ w_{\mathrm{r}}^{(k)}],\ \forall k = 1, \ldots, N_{\mathrm{c}}$ of the input variables.

Next, we will discuss how to fuse the above two types of data together to achieve the goal of improving the prediction accuracy of the original CAE model in both time domain and displacement domain.

### 3.1 Temporal convolutional network-based fusion of CAE data and test data in time domain

In the time domain, the deceleration of vehicle structural response is treated as a time series for given initial crash speed and vehicle weights. To date, there are a variety of ML techniques for the task of sequential modeling (i.e., prediction of sequential or time-series data) (Zhao et al. 2022), such as recurrent neural networks (RNN) including long short-term memory networks (LSTM; Hochreiter and

Schmidhuber 1997) and gate recurrent units (GRUs; Chung et al. 2014), and CNN (Albawi et al. 2017). More recently, Bai et al. (2018) systematically and empirically evaluated TCN which absorbs the best practices of modern CNN (e.g., dilations, completion, and residual connections) and RNN architectures on a diverse range of sequence modeling problems. The results demonstrate that TCN is substantially superior to RNN in different benchmarks and experimental tests. TCN preserves the main features of LSTM and maps a sequence of any length to an output sequence of the same length. In addition to the accuracy improvement over LSTM and CNN, TCN is faster to train and is less memory intensive. There have been numerous research corroborating the effectiveness and desirable prediction accuracy of TCN in sequence modeling tasks (Lea et al. 2016, 2017; Wan et al. 2019), which also validate that TCN outperforms than traditional RNN in terms of training efficiency, robustness to time-delays, and memory duration, and would be a promising alternative to RNN. Considering the advantages of TCN, it is employed in this paper to model the deceleration response.

Figure 3 shows a simple example of TCN architecture, which visualizes a dilated casual convolution with the dilation factors are (1, 2, 4, 8), and a filter size of 2. In TCN, a 1D fully-convolutional network is applied to realize the complete information transformation from future to past, in other words, there is no missing information in TCN-based sequence modeling. Typically, a TCN architecture consists of three main components: casual convolutions, dilated convolutions, and residual connections.

Casual convolutions allow to only convolve an output at time $t$ with elements before this moment. With casual convolution, the TCN can operate on any length of sequential input data and produce an output with the same length. Regarding dilated convolution, it aims at enlarging the receptive field that is the maximum number of steps back in time from current sample at time $t$ by applying a larger filter and magnifying dilation factor. Consequently, a large effective history in network remains. The dilated convolution is defined by

$$(F_{*d}l)(p) = \sum_{i=0}^{k-1} l(i)F_{p-di}, \tag{4}$$

where $d$ is the dilation factors, $* d$ is the $d$-dilated convolution, $k$ is the filter size, $l(i)$ is the applied filter, $F_{p-di}$ is the input, and $(F_{*d}l)(p)$ is the output.

Since dilated convolution provides a larger coverage with the same computational cost, it allows for faster training and requires less memory. The objective of residual connection is to effectively connect residual blocks, as shown in Fig. 4, which include two layers of dilated casual convolutions (the layer configuration is presented in Fig. 3), rectified linear unit (ReLU), normalization and spatial dropout. Furthermore, a residual block enables

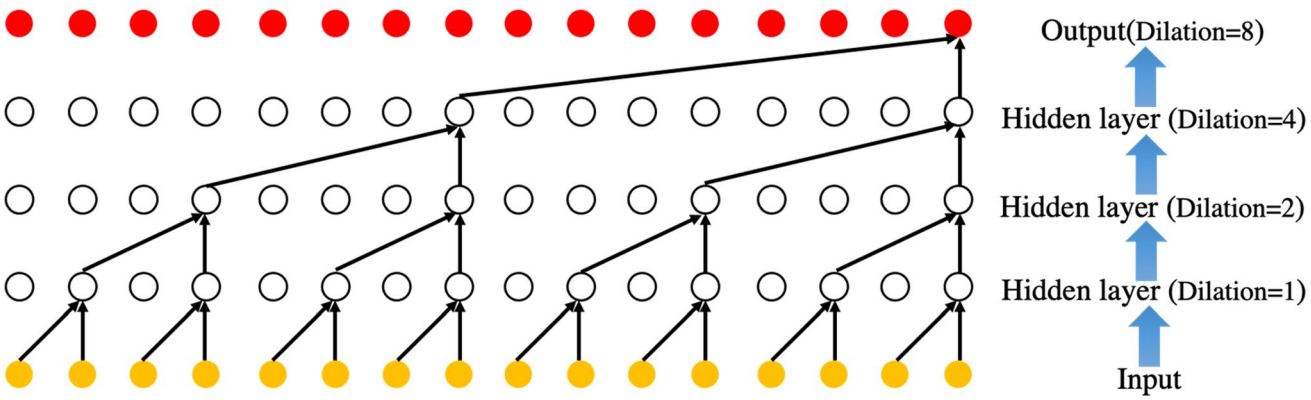


**Fig. 3** Dilated casual convolutional layer in TCN architecture

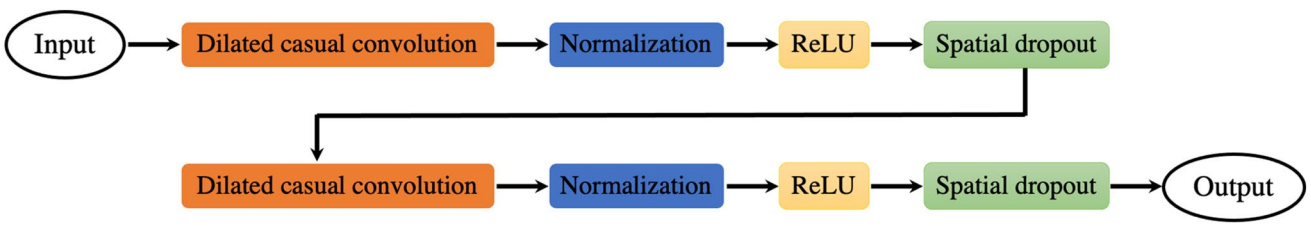


**Fig. 4** TCN residual block

to automatically adjust identity mapping and ensure the stabilization of depth and size of TCN architecture. It is also beneficial to build a very deep network with more layers and abandon some useless transformation. The generic TCN model is constructed by stacking a sequence of residual blocks and the padding in each layer is set to (kernel size − 1) multiplies by dilation factor. For detailed introduction of TCN, one can found in Bai et al. (2018).

In summary, TCN for sequence modeling tasks profits in several aspects (Bai et al. 2018): (1) the convolutions in TCN can parallelly work on a long input sequence as each layer has the same filter; (2) TCN is flexible in receptive field, memory size in model therefore can be better manipulated; (3) TCN has a stable gradient due to different backpropagation path in sequence directions, thus avoiding vanishing gradient; (4) TCN demands a lower memory to store results for training compared to RNN because of mutual filters among layers; (5) TCN can also work on sequential data with arbitrary size using 1D convolutional kernels.

To use TCN for the fusion of CAE data and test data, we treat CAE data as low-fidelity data to train an initial TCN. The initial TCN is then fine-tuned using test data (i.e., high-fidelity data) to obtain a multi-fidelity TCN architecture. For the training of the initial low-fidelity TCN using CAE data $\mathbf{a}_{\mathrm{s}}^{(j)}$ and $\mathbf{x}^{(j)} = [v_0^{(j)},\ w_{\mathrm{f}}^{(j)},\ w_{\mathrm{r}}^{(j)}],\ \forall j = 1, \ldots, N_{\mathrm{e}}$, the task is to map a sequence $\mathbf{t} = [t_1,\ t_2, \ldots, t_{N_t}]$ to another sequence $\mathbf{a}_{\mathrm{s}} = [a_{\mathrm{s},1},\ a_{\mathrm{s},2}, \ldots, a_{\mathrm{s},N_t}]$ with the same length. Note that when generating CAE data or test data, the inputs$v_0$,$w_{\mathrm{f}}$, $w_{\mathrm{r}}$ are known, the acceleration can be accordingly obtained by CAE model or actual measurement. In other words, the configuration of training data is that one set of inputs$v_0$,$w_{\mathrm{f}}$, $w_{\mathrm{r}}$ results in one set of acceleration data.

Based on the $N_e$ training samples from the CAE simulation, hyperparameters $\boldsymbol{\theta}_{\mathrm{LF}}$ of the low-fidelity TCN are obtained by minimizing the difference between predictions from the TCN model and their counterparts from the CAE model as follows

$$J_{\mathrm{LF}} = \arg\min_{\boldsymbol{\theta}_{\mathrm{LF}}} \left\{ \sum_{j=1}^{N_{\mathrm{e}}} L(\hat{\mathbf{a}}_{\mathrm{s}}(v_0^{(j)},\ w_{\mathrm{f}}^{(j)},\ w_{\mathrm{r}}^{(j)}; \boldsymbol{\theta}_{\mathrm{LF}}),\ \mathbf{a}_{\mathrm{s}}^{(j)}) \right\}, \quad (5)$$

where $\hat{\mathbf{a}}_{\mathrm{s}}(v_0^{(j)},\ w_{\mathrm{f}}^{(j)},\ w_{\mathrm{r}}^{(j)}; \boldsymbol{\theta}_{\mathrm{LF}}) = [\hat{a}_{\mathrm{s},1}^{(j)},\ \hat{a}_{\mathrm{s},2}^{(j)},\ \ldots,\ \hat{a}_{\mathrm{s},N_t}^{(j)}]$ are the predicted deceleration sequence from the low-fidelity TCN for given model inputs $\mathbf{x} = [v_0^{(j)},\ w_{\mathrm{f}}^{(j)},\ w_{\mathrm{r}}^{(j)}]$ and hyperparameter $\boldsymbol{\theta}_{\mathrm{LF}}$, $L(\cdot)$ is a loss function used in TCN training (i.e., mean square error in this paper).

The trained TCN model in Eq. (5) only uses the CAE data. Even though it can drastically reduce the required computational time for prediction, the accuracy of the CAE model has not been improved yet. To improve the prediction accuracy of the low-fidelity TCN, actual test data are then used to fine-tune the TCN model by following the multi-fidelity surrogate modeling concept and treating test data as the high-fidelity data.

Theoretically speaking, more test data could lead to a more accurate multi-fidelity TCN model. Unfortunately, collecting a large amount of test data from crash tests requires considerable effort in time and is economically very expensive. Thus, only very limited test data is available. To address this issue, transfer learning is employed to update the low-fidelity TCN model as a multi-fidelity TCN model using the limited test data. Transfer learning is a method to reuse knowledge/information from one domain in another related domain (Pan and Yang 2009; Huang et al. 2022). It substantially helps to reduce the efforts on recollecting training data. In fact, in many cases, rebuilding model from scratch and data recollection is expensive and unfeasible. It would be desirable to use transfer learning to improve learning performance in terms of cost, time, and accuracy. In this study, existed knowledge can be extracted from a pre-trained low-fidelity TCN model. The new test data is then embedded in low-fidelity TCN model to create a multi-fidelity TCN model. Let the hyperparameters of the multi-fidelity TCN model be $\boldsymbol{\theta}_{\mathrm{MF}}$, $\boldsymbol{\theta}_{\mathrm{MF}}$ are estimated using the actual test data $\mathbf{a}_{\mathrm{d}}^{(k)},\ k = 1, \ldots, N_{\mathrm{c}}$ and $\mathbf{x}^{(k)} = [v_0^{(k)},\ w_{\mathrm{f}}^{(k)},\ w_{\mathrm{r}}^{(k)}],\ \forall k = 1, \ldots, N_{\mathrm{c}}$ by solving the following optimization model

$$J_{\mathrm{MF}} = \arg\min_{\boldsymbol{\theta}_{\mathrm{MF}}} \left\{ \sum_{k=1}^{N_{\mathrm{c}}} L(\hat{\mathbf{a}}_{\mathrm{d}}(v_0^{(k)},\ w_{\mathrm{f}}^{(k)},\ w_{\mathrm{r}}^{(k)}; \boldsymbol{\theta}_{\mathrm{MF}}),\ \mathbf{a}_{\mathrm{d}}^{(k)}) \right\}, \quad (6)$$

where $\hat{\mathbf{a}}_{\mathrm{d}}(v_0^{(k)},\ w_{\mathrm{f}}^{(k)},\ w_{\mathrm{r}}^{(k)}; \boldsymbol{\theta}_{\mathrm{MF}}) = [\hat{a}_{\mathrm{d},1}^{(k)},\ \hat{a}_{\mathrm{d},2}^{(k)}, \ldots,\ \hat{a}_{\mathrm{d},N_t}^{(k)}]$ are the predicted deceleration sequence from multi-fidelity TCN for given inputs $v_0^{(k)},\ w_{\mathrm{f}}^{(k)},\ w_{\mathrm{r}}^{(k)}$ and hyperparameters $\boldsymbol{\theta}_{\mathrm{MF}}$, $\mathbf{a}_{\mathrm{d}}^{(k)}$ is the true deceleration sequence of the $k$th vehicle crash test, and $L(\cdot)$ is a loss function as mentioned in Eq. (5). With transfer learning, in multi-fidelity TCN model, $\boldsymbol{\theta}_{\mathrm{MF}} = \boldsymbol{\theta}_{\mathrm{LF}} + \boldsymbol{\theta}_{\mathrm{HF}}$. $\boldsymbol{\theta}_{\mathrm{LF}}$ are hyperparameters of the low-fidelity TCN model which have been optimized using Eq. (5) and considered as existing knowledge. Thus, they can be frozen when training the multi-fidelity TCN. $\boldsymbol{\theta}_{\mathrm{HF}}$ are additive hyperparameters resulting from test data $\mathbf{a}_{\mathrm{d}}^{(\mathrm{k})},\ k = 1, \ldots, N_{\mathrm{c}}$, which targets on modifying $\boldsymbol{\theta}_{LF}$ to transfer the low-fidelity TCN model to a multi-fidelity TCN model using available test data. More specifically, as a multi-fidelity TCN model is created based on current predictive capability in the pre-trained low-fidelity TCN and available test data, new hyperparameters $\boldsymbol{\theta}_{\mathrm{MF}}$ of multi-fidelity TCN in Eq. (6) are optimized based on $\boldsymbol{\theta}_{\mathrm{LF}}$ in low-fidelity TCN using actual test data. The difference $\boldsymbol{\theta}_{\mathrm{HF}}$ between $\boldsymbol{\theta}_{\mathrm{LF}}$ and $\boldsymbol{\theta}_{\mathrm{MF}}$ arises from the use of test data and allows to fine-tune low-fidelity TCN into

multi-fidelity TCN. Therefore, $\boldsymbol{\theta}_{\mathrm{LF}}$ can be considered as the initial values upon optimization of $\boldsymbol{\theta}_{\mathrm{MF}}$.

Figure 5 illustrates the flowchart of fusion CAE data and test data using transfer learning and TCN in time domain. In the first place, a low-fidelity TCN model is trained using only CAE data by Eq. (5). Due to limited test data available, the transfer learning is then adopted to update the low-fidelity TCN model to a multi-fidelity TCN model using a small amount of test data. During transfer process, hyperparameters of the pre-trained low-fidelity TCN model are frozen and modified by additional hyperparameters arising from test data. Finally, validation on the trained multi-fidelity TCN model is performed using extra test data. It is seen in Fig. 5 that the fusion of the CAE data and test data in time domain is realized by directly tuning hyperparameters in ML-based models (herein TCN model) in order to minimize the mismatch between model prediction and real-world observation, which is intuitively simple and understandable.

### 3.2 Fusion of CAE data and test data in the displacement domain

Although the time-domain approach is straightforward for the fusion of CAE data and test data, the resulting multi-fidelity TCN model is still difficult to interpolate due to the highly complicated architecture of the neural networks. In this section, a displacement-domain approach for model bias correction is proposed. First, we assume that a vehicle crash test can be modeled as a spring-mass system,

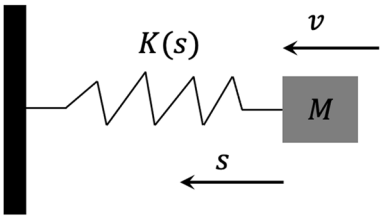


**Fig. 6** Spring–mass model for vehicle crash test

as shown in Fig. 6. Based on the fundamental structural dynamics, the dynamic motion representing Fig. 6 can be characterized by Eqs. (7) and (8) as follows

$$M\frac{\mathrm{d}v}{\mathrm{d}t} = -K(s)s(t), \tag{7}$$

or

$$M\mathrm{d}v = -K(s)s(t)\mathrm{d}t, \tag{8}$$

where $M$ is the vehicle mass; $v$ is the vehicle speed at the time moment $t$; $K(s)$ is a nonlinear spring constant as a function of displacement $s$, and $s(t)$ is a displacement function with respect to $t$.

To solve differential equations (7) or (8), the numerical integration is used to estimate dynamic properties, e.g., displacement, deceleration, and velocity, by defining initial displacement and deceleration as zero, and initial velocity as $v_0$, which are given by

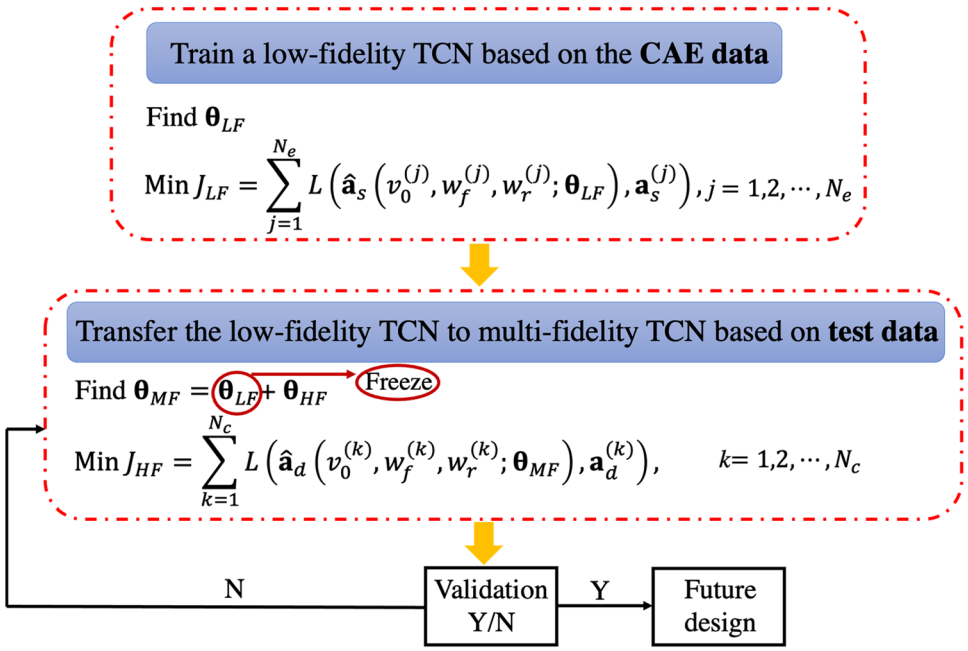


**Fig. 5** Model bias correction in time domain

$$
\begin{aligned}
&i = 0, \quad s_0 = 0, \quad v_0 = v_0, \quad a_0 = 0,\\
&\text{For } i = 0 \text{ to } N_t:\\
&i = i + 1,\\
&s_{i+1} = s_i + v_i \Delta t,\\
&v_{i+1} = v_i + a_i \Delta t,\\
&a_{i+1} = \underbrace{-\frac{1}{M} K(s_i) s_i}_{\text{Unknow or unable to model accurately}},
\end{aligned}
\tag{9}
$$

where $\Delta t$ is the time step of discretization in time and $N_t$ is the total number of steps.

It is seen in Eq. (9) that deceleration is controlled by $K(s)$, which is highly nonlinear and cannot be accurately determined through theory due to the highly complicated structure of the vehicle. The dynamic properties, therefore, cannot be directly obtained by solving differential equations. To capture the unknown function of the nonlinear spring constant as a function of displacement, we leverage both the capacity of CAE simulation and ML model in the displacement domain and approximate $a_{i+1}$ in Eq. (9) as the following function

$$
a_{i+1} = G_a(s_i,\ v_0,\ w_\mathrm{f},\ w_\mathrm{r}) + \delta(s_i,\ v_0,\ w_\mathrm{f},\ w_\mathrm{r}),
\tag{10}
$$

where $G_a(s_i,\ v_0,\ w_\mathrm{f},\ w_\mathrm{r})$ is the CAE model which predicts the deceleration for given displacement $s_i$ and $v_0$, $w_\mathrm{f}$, $w_\mathrm{r}$, and $\delta(s_i,\ v_0,\ w_\mathrm{f},\ w_\mathrm{r})$ is a ML model that compensates the unmodeled deceleration as a function of displacement by the CAE model. It is worth mentioning that $G_a(s_i,\ v_0,\ w_\mathrm{f},\ w_\mathrm{r})$ is the CAE predictive model in the displacement domain, which is different from the time-domain model given in Eq. (3).

To train a ML model for $\delta(s_i,\ v_0,\ w_\mathrm{f},\ w_\mathrm{r})$, training data need to be identified first in the displacement domain. We therefore first convert the time-domain data given at the beginning of Sect. 3 into data in the displacement domain and then obtain data necessary for the training of $\delta(s_i,\ v_0,\ w_\mathrm{f},\ w_\mathrm{r})$. For the $k$th crash test, where $k = 1, \ldots, N_\mathrm{c}$, the test condition is represented by input variables $\mathbf{x}^{(k)} = [v_0^{(k)},\ w_\mathrm{f}^{(k)},\ w_\mathrm{r}^{(k)}]$, the corresponding deceleration sequence is $\mathbf{a}_\mathrm{d}^{(k)} = [a_{\mathrm{d},1}^{(k)},\ a_{\mathrm{d},2}^{(k)}, \ldots, a_{\mathrm{d},N_t}^{(k)}],\ \forall k = 1, \ldots, N_\mathrm{c}$. Define the initial velocity of the $k$th crash test as $v_{\mathrm{d},0}^{(k)} = v_0^{(k)}$, based on the fundamental dynamic analysis for $N_t$ time steps, we have

$$
\begin{aligned}
&s_{\mathrm{d},0}^{(k)} = 0; \quad v_{\mathrm{d},0}^{(k)} = v_0^{(k)};\\
&\text{For } i = 1 \text{ to } N_t:\\
&s_{d,i}^{(k)} = s_{d,i-1}^{(k)} + v_{d,i-1}^{(k)} \Delta t,\\
&v_{d,i}^{(k)} = v_{d,i-1}^{(k)} + a_{d,i-1}^{(k)} \Delta t,
\end{aligned}
\tag{11}
$$

where $s_{d,i}^{(k)},\ i = 1, \ldots, N_t$ is the displacement of the $k$th crash test at time step $t_i$.

Based on Eq. (11), we can easily transform the time-domain crash test data into displacement domain as

$$
\begin{aligned}
\mathbf{s}_\mathrm{d}^{(k)} &= [s_{\mathrm{d},1}^{(k)}, \ldots, s_{\mathrm{d},N_t}^{(k)}];\ \mathbf{v}_\mathrm{d}^{(k)}\\
&= [v_{\mathrm{d},0}^{(k)},\ v_{\mathrm{d},1}^{(k)}, \ldots, v_{\mathrm{d},N_t}^{(k)}];\ \mathbf{a}_\mathrm{d}^{(k)}\\
&= [a_{\mathrm{d},1}^{(k)}, \ldots, a_{\mathrm{d},N_t}^{(k)}];\ \forall k = 1, \ldots, N_\mathrm{c}.
\end{aligned}
\tag{12}
$$

Using $\mathbf{x}^{(k)} = [v_0^{(k)},\ w_\mathrm{f}^{(k)},\ w_\mathrm{r}^{(k)}]$ of the $k$th crash test as inputs and the CAE model, we obtain the corresponding deceleration sequence from the CAE model as $\mathbf{a}_\mathrm{s}^{(k)} = G_\mathrm{s}(\mathbf{x}^{(k)}) = G_\mathrm{s}(v_0^{(k)},\ w_\mathrm{f}^{(k)},\ w_\mathrm{r}^{(k)})$. By carrying out a similar procedure as Eq. (11) for the CAE data, $\mathbf{a}_s^{(k)}$, we have

$$
\begin{aligned}
&s_{\mathrm{s},0}^{(k)} = 0; \quad v_{\mathrm{s},0}^{(k)} = v_0^{(k)};\\
&\text{For } i = 1 \text{ to } N_t:\\
&s_{s,i}^{(k)} = s_{s,i-1}^{(k)} + v_{s,i-1}^{(k)} \Delta t,\\
&v_{s,i}^{(k)} = v_{s,i-1}^{(k)} + a_{s,i-1}^{(k)} \Delta t.
\end{aligned}
\tag{13}
$$

Finally, we have the deceleration data in the displacement from the CAE model corresponding to the $k$th crash test condition as

$$
\begin{aligned}
\mathbf{s}_\mathrm{s}^{(k)} &= [s_{\mathrm{s},0}^{(k)},\ s_{\mathrm{s},1}^{(k)}, \ldots,\ s_{\mathrm{s},N_t}^{(k)}];\ \mathbf{v}_\mathrm{s}^{(k)}\\
&= [v_{\mathrm{s},0}^{(k)},\ v_{\mathrm{s},1}^{(k)}, \ldots, v_{\mathrm{s},N_t}^{(k)}];\ \mathbf{a}_\mathrm{s}^{(k)}\\
&= [a_{\mathrm{s},1}^{(k)}, \ldots, a_{\mathrm{s},N_t}^{(k)}];\ \forall k = 1, \ldots, N_\mathrm{c}.
\end{aligned}
\tag{14}
$$

It is worth noting that in the time domain (time vs. deceleration), data points in CAE simulation and actual test have one-to-one mapping relationship during the same time period, since each time step is considered as input and static. However, after the conversion of time domain to displacement domain (displacement vs. deceleration), each data point in CAE simulation and actual test may not necessarily maintain one-to-one mapping due to numerical integration with respect to time in differential equations, as illustrated in Fig. 7. As a result, we cannot obtain the required model bias data for the training of $\delta(s_i,\ v_0,\ w_\mathrm{f},\ w_\mathrm{r})$ by directly subtracting data given in Eq. (14) from that in Eq. (12). Or in other words,

$$
\delta(s_{s,i}^{(k)},\ v_0^{(k)},\ w_\mathrm{f}^{(k)},\ w_\mathrm{r}^{(k)}) \neq a_{d,i}^{(k)} - a_{s,i}^{(k)}; \quad \forall k = 1, \ldots, N_\mathrm{c}; \quad i = 1, \ldots, N_t.
\tag{15}
$$

To achieve a one-to-one mapping in displacement domain and further capture model bias between CAE prediction and actual test, a bijective projection scheme is developed. Two scenarios (see Fig. 7) are considered to realize the bijective

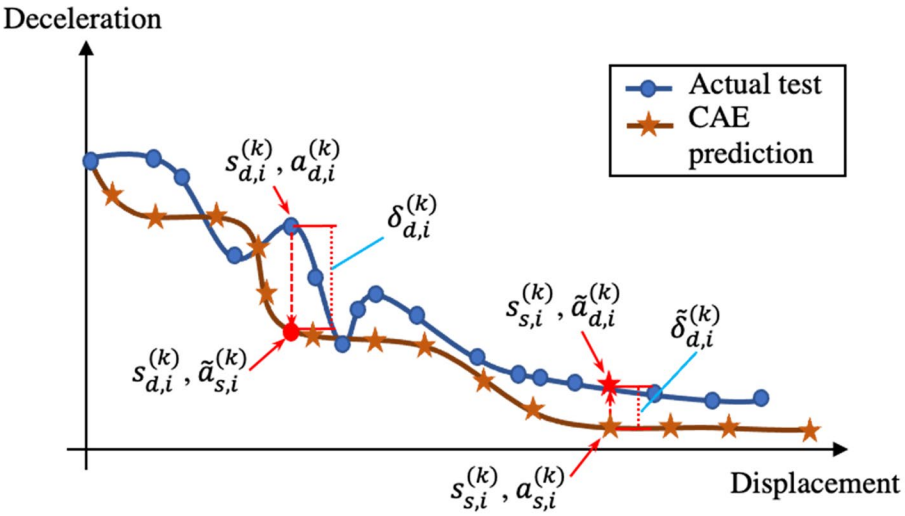


**Fig. 7** Model bias analysis in the displacement domain

projection: (1) projection from CAE prediction to actual test; (2) projection from actual test to CAE prediction.

More specifically, for the projection of CAE prediction to test, the interpolation is implemented using CAE data. Therefore, the estimated deceleration from the CAE model for the $k$th crash test at time step $t_i$ is expressed as

$$\tilde{a}_{s,i}^{(k)} = f_{\text{int}}(s_{d,i}^{(k)}; \mathbf{s}_{\text{s}}^{(k)}, \mathbf{a}_{\text{s}}^{(k)}), \quad \forall k = 1, \ldots, N_{\text{c}}; \quad i = 1, \ldots, N_t, \tag{16}$$

where $f_{\text{int}}(s_{d,i}^{(k)}; \mathbf{s}_{\text{s}}^{(k)}, \mathbf{a}_{\text{s}}^{(k)})$ is an interpolation function to predict response of $s_{d,i}^{(k)}$ based on data $\mathbf{s}_{\text{s}}^{(k)}$, $\mathbf{a}_{\text{s}}^{(k)}$ given in Eq. (14) using interpolation.

The model bias is then calculated using the estimated CAE response $\tilde{a}_{s,i}^{(k)}$ and corresponding test response $a_{d,i}^{(k)}$ as

$$\delta_{d,i}^{(k)} = a_{d,i}^{(k)} - \tilde{a}_{s,i}^{(k)}. \tag{17}$$

Keep calculating model bias for all $s_{d,i}^{(k)}$ from the $N_{\text{c}}$ groups of deceleration data at $N_t$ time steps, we have all inputs and bias (outputs) from this scenario of projection as follows

$$[\mathbf{x}_{\text{d}}; \boldsymbol{\delta}_{\text{d}}] = \begin{cases} \text{Inputs: } \mathbf{x}_{d,i}^{(k)} = [s_{d,i}^{(k)}, v_0^{(k)}, w_{\text{f}}^{(k)}, w_{\text{r}}^{(k)}] \\ \text{Bias: } \delta_{d,i}^{(k)}, \end{cases}, \quad \forall k = 1, \ldots, N_{\text{c}}; \quad i = 1, \ldots, N_t. \tag{18}$$

Similarly, for the scenario of projecting CAE data to test data as illustrated in Fig. 7, the estimated deceleration corresponding to $x_{s,i}^{(k)}$ is expressed as

$$\tilde{a}_{d,i}^{(k)} = f_{\text{int}}(s_{s,i}^{(k)}; \mathbf{x}_{\text{d}}^{(k)}, \mathbf{a}_{\text{d}}^{(k)}), \tag{19}$$

where $f_{\text{int}}(s_{s,i}^{(k)}; \mathbf{x}_{\text{d}}^{(k)}, \mathbf{a}_{\text{d}}^{(k)})$ is an interpolation function for the estimation of deceleration for $x_{s,i}^{(k)}$ using test data $\mathbf{x}_{\text{d}}^{(k)}$, $\mathbf{a}_{\text{d}}^{(k)}$.

Based on the estimation given in Eq. (19), the model bias of deceleration corresponds to $x_{s,i}^{(k)}$ is calculated as

$$\tilde{\delta}_{d,i}^{(i)} = \tilde{a}_{d,i}^{(k)} - a_{s,i}^{(k)}. \tag{20}$$

Using Eqs. (19) and (20), we have all inputs and bias outputs for $s_{s,i}^{(k)}$ $\forall k = 1, \ldots, N_{\text{c}}$; $i = 1, \ldots, N_t$ as

$$[\mathbf{x}_{\text{s}}; \boldsymbol{\delta}_{\text{s}}] = \begin{cases} \text{Inputs: } \mathbf{x}_{s,i}^{(k)} = [s_{s,i}^{(k)}, v_0^{(k)}, w_{\text{f}}^{(k)}, w_{\text{r}}^{(k)}] \\ \text{Bias: } \tilde{\delta}_{d,i}^{(k)} \end{cases}, \quad \forall k = 1, \ldots, N_{\text{c}}; \quad i = 1, \ldots, N_t. \tag{21}$$

After that, we can combine data obtained from the two scenarios and obtain data needed for the training of $\delta(s_i, v_0, w_{\text{f}}, w_{\text{r}})$ as

$$\begin{aligned} \mathbf{x}_{\delta,t} &= \{\mathbf{x}_{\text{d}} \cup \mathbf{x}_{\text{s}}\}, \\ \boldsymbol{\delta}_t &= \{\boldsymbol{\delta}_{\text{d}} \cup \boldsymbol{\delta}_{\text{s}}\}. \end{aligned} \tag{22}$$

Once the training data $\mathbf{x}_{\delta,t}$ and $\boldsymbol{\delta}_t$, including all inputs and bias outputs, are available, a ML model can be constructed to represent the relations between the bias of deceleration and inputs including displacement, mass, and initial crash speed. In this paper, GPR is employed in the displacement domain to model this unknown relationship. The GPR model has been widely used as a surrogate model of computationally expensive simulators. In essence, GPR can be generalized as a Gaussian process with a Bayesian formulation. It has strong ability to perform interpolation, even if observations are noise-contaminated (Rasmussen 2003). In addition, the GPR is quite flexible as it enables to emulate a wide range of functional forms. The mechanism in GPR also allows to quantify the error in the prediction in the form of covariance.

In GPR, the output $y_\delta = \delta(s, v_0, w_{\text{f}}, w_{\text{r}})$ is approximated by a Gaussian process $\hat{G}_\delta(\mathbf{x}_\delta)$, where $\mathbf{x}_\delta = [s, v_0, w_{\text{f}}, w_{\text{r}}]$ with mean and covariance given by

$$E\left[\hat{G}_\delta(\mathbf{x}_\delta)\right] = \mathbf{h}^T(\mathbf{x}_\delta)\boldsymbol{\beta},$$
$$\mathrm{Cov}\left[\hat{G}_\delta(\mathbf{x}_\delta),\ \hat{G}_\delta(\mathbf{x'}_\delta)\right] = \sigma^2 R(\mathbf{x}_\delta,\ \mathbf{x'}_\delta), \tag{23}$$

where $\mathbf{h}^T(\mathbf{x}_\delta)$ is a vector of known regression functions, e.g., constant or linear function, $\boldsymbol{\beta}$ is a column vector of coefficients that are related to polynomial regression of $\mathbf{h}^{\mathrm{T}}(\mathbf{x}_\delta)$, $\sigma^2$ is a constant, and $R(\mathbf{x}_\delta,\ \mathbf{x'}_\delta)$ is a correlation function between the responses at two points $\mathbf{x}_\delta$ and $\mathbf{x'}_\delta$ (Schulz et al. 2018).

A widely used correlation function of $R(\mathbf{x}_\delta,\ \mathbf{x'}_\delta)$ is given by Quinonero-Candela and Rasmussen (2005)

$$R(\mathbf{x}_\delta,\ \mathbf{x'}_\delta) = \prod_{i=1}^{4} \exp\left[\rho_i (x_{\delta,i} - x'_{\delta,i})^2\right], \tag{24}$$

where $\boldsymbol{\rho} = [\rho_1,\ \rho_2,\ \rho_3,\ \rho_4]^{\mathrm{T}}$ is a vector of correlation parameters representing the change in correlation between $\hat{G}_\delta(\mathbf{x}_\delta)$ and $\hat{G}_\delta(\mathbf{x'}_\delta)$ with the difference of $\mathbf{x}_\delta$ and $\mathbf{x'}_\delta$ increasing. Based on the derivations of GPR, the hyperparameters $\boldsymbol{\beta}$, $\sigma^2$, and $\boldsymbol{\rho}$ can completely characterize the GPR, which can be solved using the maximum likelihood estimation (MLE) method (Park and Jeon 2002).

As the response $\hat{G}_\delta(\mathbf{x}_\delta)$ is assumed as Gaussian process with mean $\mathbf{h}^{\mathrm{T}}(\mathbf{x}_\delta)\boldsymbol{\beta}$ and covariance matrix $\sigma^2 R(\mathbf{x}_\delta,\ \mathbf{x'}_\delta)$, the likelihood function of observing $\boldsymbol{\delta}_t$ is given by

$$L_{\mathrm{GP}}(\boldsymbol{\beta}, \sigma, \boldsymbol{\rho}) = (2\pi\sigma^2)^{-\frac{N_\delta}{2}} |\mathbf{R}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{\delta}_t - \mathbf{H}\boldsymbol{\beta})^{\mathrm{T}}\mathbf{R}^{-1}(\boldsymbol{\delta}_t - \mathbf{H}\boldsymbol{\beta})\right], \tag{25}$$

where $N_\delta$ is the number of points in training data $\mathbf{x}_{\delta,t}$ and $\boldsymbol{\delta}_t$ obtained in Eqs. (16) through (22), $\mathbf{H} = [\mathbf{h}^{\mathrm{T}}(\mathbf{x}_{\delta,t}^{(1)}),\ \mathbf{h}^{\mathrm{T}}(\mathbf{x}_{\delta,t}^{(2)}), \ldots, \mathbf{h}^{\mathrm{T}}(\mathbf{x}_{\delta,t}^{(N_\delta)})]^{\mathrm{T}}$, $\mathbf{R}$ is a $N_\delta \times N_\delta$ matrix whose elements are $R(\mathbf{x}_{\delta,t}^{(i)},\ \mathbf{x}_{\delta,t}^{(j)}),\ \forall i,\ j = 1, \ldots, N_\delta$, in which $\mathbf{x}_{\delta,t}^{(i)}$ is the $i$th sample in $\mathbf{x}_{\delta,t}$.

Based on the likelihood function given in Eq. (25), the MLE $\hat{\boldsymbol{\beta}}$, $\hat{\sigma}^2$, and $\hat{\boldsymbol{\rho}}$ of hyperparameters $\boldsymbol{\beta}$, $\sigma^2$, and $\boldsymbol{\rho}$ can be obtained. After $\hat{\boldsymbol{\beta}}$, $\hat{\sigma}^2$, and $\hat{\boldsymbol{\rho}}$ are obtained, for a new point $\mathbf{x}_\delta^{\mathrm{new}} = [s^{\mathrm{new}},\ v_0^{\mathrm{new}},\ w_{\mathrm{f}}^{\mathrm{new}},\ w_{\mathrm{r}}^{\mathrm{new}}]$, we have the prediction as

$$\hat{G}_\delta(\mathbf{x}_\delta^{\mathrm{new}}) \sim N(\mu_\delta(\mathbf{x}_\delta^{\mathrm{new}}),\ \sigma_\delta^2(\mathbf{x}_\delta^{\mathrm{new}})), \tag{26}$$

where $N(\cdot, \cdot)$ is a normal distribution, $\mu_\delta(\mathbf{x}_\delta^{\mathrm{new}})$ and $\sigma_\delta^2(\mathbf{x}_\delta^{\mathrm{new}})$ are given by

$$\mu_\delta(\mathbf{x}_\delta^{\mathrm{new}}) = \mathbf{h}^{\mathrm{T}}(\mathbf{x}_\delta^{\mathrm{new}})\hat{\boldsymbol{\beta}} + \mathbf{r}^{\mathrm{T}}(\mathbf{x}_\delta^{\mathrm{new}})\mathbf{R}^{-1}(\boldsymbol{\delta}_t - \mathbf{H}\hat{\boldsymbol{\beta}}),$$
$$\sigma_\delta^2(\mathbf{x}_\delta^{\mathrm{new}}) = \sigma^2 \left\{ \begin{array}{l} 1 - \mathbf{r}^{\mathrm{T}}(\mathbf{x}_\delta^{\mathrm{new}})\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}_\delta^{\mathrm{new}}) \\ +[\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}_\delta^{\mathrm{new}}) - \boldsymbol{\delta}_t]^{\mathrm{T}}(\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H})[\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}_\delta^{\mathrm{new}}) - \boldsymbol{\delta}_t] \end{array} \right\}, \tag{27}$$

in which $\mathbf{r}^{\mathrm{T}}(\mathbf{x}_\delta^{\mathrm{new}})$ is a $N_\delta \times 1$ vector, of which the $i$th element is $R(\mathbf{x}_\delta^{\mathrm{new}},\ \mathbf{x}_{\delta,t}^{(i)}),\ \forall i = 1, \ldots, N_\delta$. More details for GPR method refer to Sobester et al. (2008) and Rasmussen and Nickisch 2010).

After the construction of the GPR model, we can approximate the unknown nonlinear function of deceleration given in Eq. (10) as

$$a = G_a(s,\ v_0,\ w_{\mathrm{f}},\ w_{\mathrm{r}}) + \hat{G}_\delta(\mathbf{x}_\delta^{\mathrm{new}}). \tag{28}$$

For any new vehicle speed $v_0^{\mathrm{new}}$ and new masses, $w_{\mathrm{f}}^{\mathrm{new}}$, $w_{\mathrm{r}}^{\mathrm{new}}$, the deceleration can be computed iteratively according to dynamic analysis as follows

$$\begin{aligned} &i = 0, \quad s_i^{\mathrm{new}} = 0, \quad v_i^{\mathrm{new}} = v_0^{\mathrm{new}}, \quad a_i^{\mathrm{new}} = 0, \\ &\text{For } i = 1 \text{ to } N_t: \\ &s_i^{\mathrm{new}} = s_{i-1}^{\mathrm{new}} + v_{i-1}^{\mathrm{new}}\Delta t, \\ &\mathbf{x}_\delta^{\mathrm{new}} = [s_i^{\mathrm{new}},\ v_0^{\mathrm{new}},\ w_{\mathrm{f}}^{\mathrm{new}},\ w_{\mathrm{r}}^{\mathrm{new}}], \\ &a_i^{\mathrm{new}} = G_a(s_i^{\mathrm{new}},\ v_0^{\mathrm{new}},\ w_{\mathrm{f}}^{\mathrm{new}},\ w_{\mathrm{r}}^{\mathrm{new}}) + \hat{G}_\delta(\mathbf{x}_\delta^{\mathrm{new}}), \\ &v_i^{\mathrm{new}} = v_{i-1}^{\mathrm{new}} + a_i^{\mathrm{new}}\Delta t. \end{aligned} \tag{29}$$

Note that $\hat{G}_\delta(\mathbf{x}_\delta^{\mathrm{new}})$ is a random variable as given in Eq. (26). In order to account for the uncertainty in the prediction, Monte Carlo simulation method needs to be performed iteratively according to the dynamic analysis scheme given in Eq. (29). Figure 8 shows the process of model training and prediction in displacement domain approach. In the first place, the vehicle crash is modeled as a spring-mass system. To capture the missing physics resulting from nonlinear spring constant, the deceleration responses with respect to time in CAE model and actual tests are first converted to those with respect to displacement. A bijective projection is developed to ensure the one-to-one mapping and to calculate model bias between CAE model and actual tests. When all inputs and bias outputs are available, a GPR model is constructed to represent the relation between model bias and inputs. The constructed GPR model allows for estimating model bias of CAE model prediction of nonlinear spring constant for any given new inputs. The model prediction starts with the initialization of displacement and acceleration and a new speed. Given a displacement at each time step, a deceleration without bias correction from CAE model is first obtained. The model bias is then estimated using the GPR model and is added to deceleration responses for deceleration model bias correction. Therefore, in the entire prediction process, it keeps updating the displacement, velocity, and acceleration in an iterative manner.

Sections 3.1 and 3.2 elaborately introduce the proposed two approaches for model bias correction, namely, time domain-based and displacement domain-based approaches. However, the working mechanisms in two approaches are different. Time domain-based approach employs transfer learning to fine-tune a low-fidelity TCN as a multi-fidelity TCN, the model bias is directly incorporated in the model by minimizing the discrepancy between model prediction and test data. In addition, the time is treated as an input and the objective is to solve a static

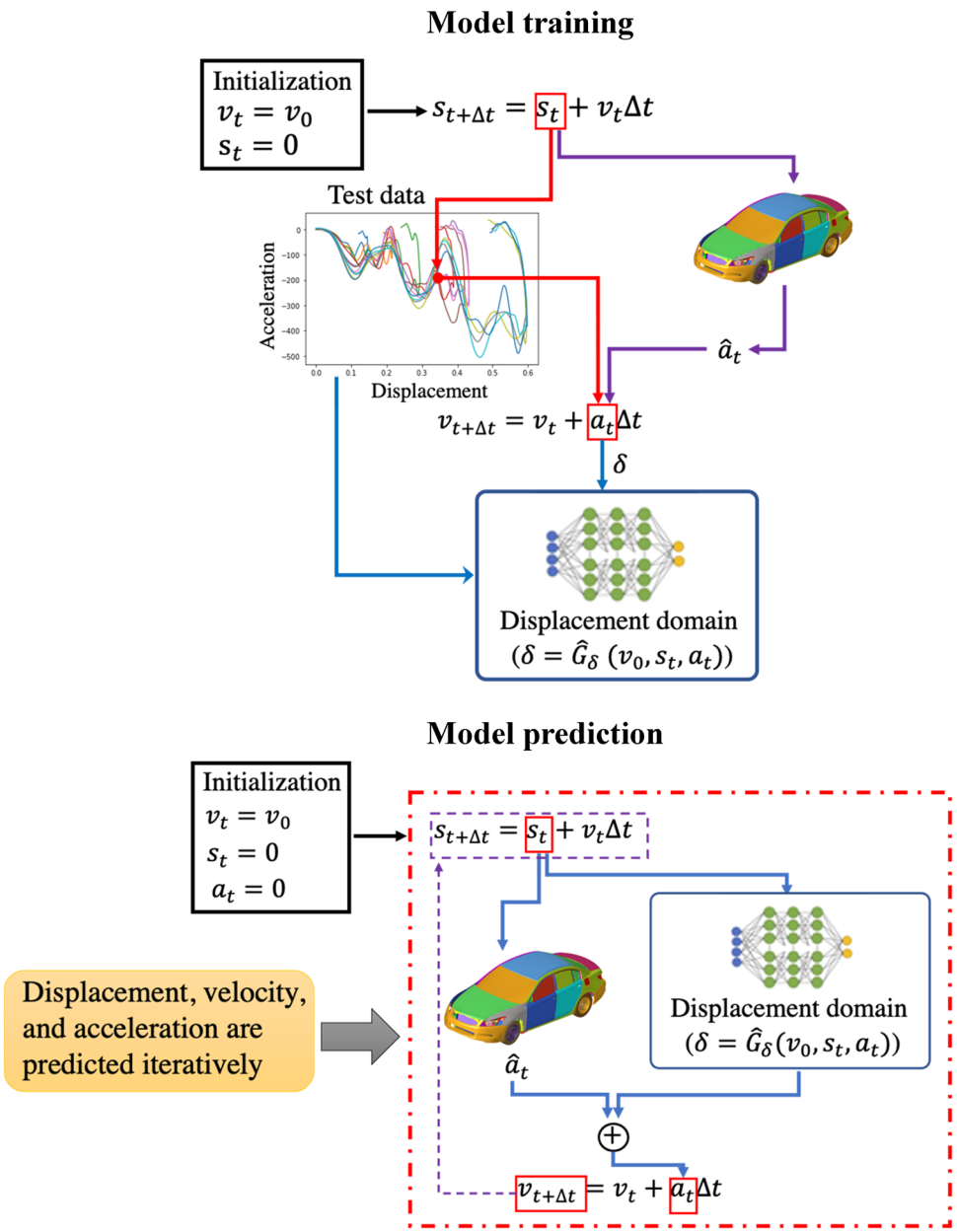


**Fig. 8** Fusion of CAE data and test data in the displacement domain

modeling problem. Displacement domain-based approach applies GPR to build a bias function, the learned GPR model is integrated into vehicle deceleration dynamic analysis along with the original CAE model to predict vehicle deceleration under a new crash speed. Uncertainty on model prediction can also be quantified due to the probabilistic prediction of GPR. In addition, there is no need to solve differential equations in time domain-based approach. While displacement domain-based approach requires to iteratively solve differential equations as it treats the overall problem as a dynamic analysis problem.

### 3.3 Vehicle crashworthiness prediction model validation using an ISO metric

Model validation is a process to determine how much a real-world system can be accurately represented by a model (Schwer 2007). Model validation is performed based on selected validation metrics. A good validation metric should quantitatively measure the correlation/discrepancy between test data and model prediction. Sarin et al. (2008) comprehensively introduced the existing validation metrics, including their benefits and drawbacks. In this study, as our responses are time series data, we adopt ISO (International Organization for Standardization) metric to assess the discrepancy between computational and experimental data. The

ISO metric takes advantage of different metrics to reliably and robustly assess quantify the degree of agreement of two time series data (Barbat et al. 2013). The overall ISO metric rating contains corridor score, phase score, magnitude score, and slope score.

Figure 9 displays the structure of ISO validation metric. The corridor score quantifies the deviation between two time series data using corridor fitting. It has the inner and outer corridors. The basic idea in this metric is that a narrow inner corridor and a wide outer corridor wide stands for a good correlation and are likely to separate a fair correlation from a poor one. Corridor score for the correlation between the true time series data and predicted counterparts can be calculated using Eqs. (30)–(31) as follows (Barbat et al. 2013)

$$\mathrm{Corr}(t) = \begin{cases} 1 & \text{if } |A(t) - B(t)| < \delta_i, \\ \left(\frac{\delta_0 - |A(t)-B(t)|}{\delta_0 - \delta_i}\right)^{k_z} & k_z \in N_t, \\ 0 & \text{if } |A(t) - B(t)| > \delta_0, \end{cases} \tag{30}$$

where $A$ and $B$ are true time series data and predicted ones at every evaluation time $t$, respectively. $\delta_i$ and $\delta_0$ are the absolute half width of the inner and outer corridors, respectively, $N_t$ is the total number of sample points. $k_z$ is a parameter that reflects data location within the outer corridor and can be defined depending on specific regression problem, such as $k_z = 1,\ 2,\ 3$ for linear, quadratic and cubic regression relation, respectively.

All Corr($t$) scores at each time step are averaged to calculate the final corridor score, which is given by

$$\mathrm{Corr} = \frac{\sum_{t=t_{\mathrm{start}}}^{t_{\mathrm{end}}} \mathrm{Corr}(t)}{N_t}. \tag{31}$$

The rest of three scores are considered as objective ratings. The phase score aims to measure the phase lag between two data. The perfect phase score is 1, indicating shifting simulation data to the maximum cross correlation is not needed. The formulation of phase score $Q_{\mathrm{p}}$ is given as follows (Barbat et al. 2013)

$$Q_{\mathrm{p}} = \begin{cases} 1 & \text{if } n_\varepsilon = 0, \\ \left(\frac{\varepsilon_{\mathrm{p}}^* \cdot N_t - n_\varepsilon}{\varepsilon_{\mathrm{p}}^* \cdot N_t}\right)^{k_{\mathrm{p}}} & k_{\mathrm{p}} \in \{1,\ 2,\ 3\}, \\ 0 & \text{if } n_\varepsilon \geq \varepsilon_{\mathrm{p}}^* \cdot N_t \end{cases} \tag{32}$$

where $n_\varepsilon$ is the phase error; $\varepsilon_{\mathrm{p}}^*$ is the maximum allowable time shift; $k_{\mathrm{p}}$ is the exponent factor that can be selected as 1 (linear), 2 (quadratic), 3 (cubic).

The magnitude score, as its name suggests, is used to measure the difference in amplitude on every single sampling point between two time series data. The magnitude score $Q_{\mathrm{m}}$ is calculated using Eq. (33) (Barbat et al. 2013)

$$Q_{\mathrm{m}} = \begin{cases} 1 & \text{if } \varepsilon_{\mathrm{mag}} = 0, \\ \left(\frac{\varepsilon_{\mathrm{m}}^* - \varepsilon_{\mathrm{mag}}}{\varepsilon_{\mathrm{m}}^*}\right)^{k_{\mathrm{m}}} & k_{\mathrm{m}} \in \{1,\ 2,\ 3\}, \\ 0 & \text{if } \varepsilon_{\mathrm{m}}^* \geq \varepsilon_{\mathrm{mag}}, \end{cases} \tag{33}$$

where $\varepsilon_{\mathrm{mag}}$ is the magnitude error. $\varepsilon_{\mathrm{m}}^*$ is the maximum allowable magnitude error. Similar to $k_{\mathrm{p}}$ in Eq. (32), $k_{\mathrm{m}}$ defines the order of regression.

The slope score targets on measuring difference in slope or the curve shape (e.g., peak or valley amounts) of two time series data. The slope at time series data refers to the slope at each point. The slope score $Q_{\mathrm{s}}$ is formulated as follows (Barbat et al. 2013)

$$Q_{\mathrm{s}} = \begin{cases} 1 & \text{if } \varepsilon_{\mathrm{slope}} = 0, \\ \left(\frac{\varepsilon_{\mathrm{s}}^* - \varepsilon_{\mathrm{slope}}}{\varepsilon_{\mathrm{s}}^*}\right)^{k_{\mathrm{s}}} & k_{\mathrm{s}} \in \{1,\ 2,\ 3\}, \\ 0 & \text{if } \varepsilon_{\mathrm{s}}^* \geq \varepsilon_{\mathrm{slope}}, \end{cases} \tag{34}$$

where $\varepsilon_{\mathrm{slope}}$ is the slope error. $\varepsilon_{\mathrm{m}}^*$ is the maximum allowable slope error. $k_{\mathrm{s}}$ is the order of regression, which is selected from 1, 2, 3.

It is worth mentioning that with concurrent use of phase, magnitude, and slope metrics, the correlation quantification between two data would be challenging due to a strong interaction among them. Therefore, a numerical optimization method is used to calculate these three scores. The

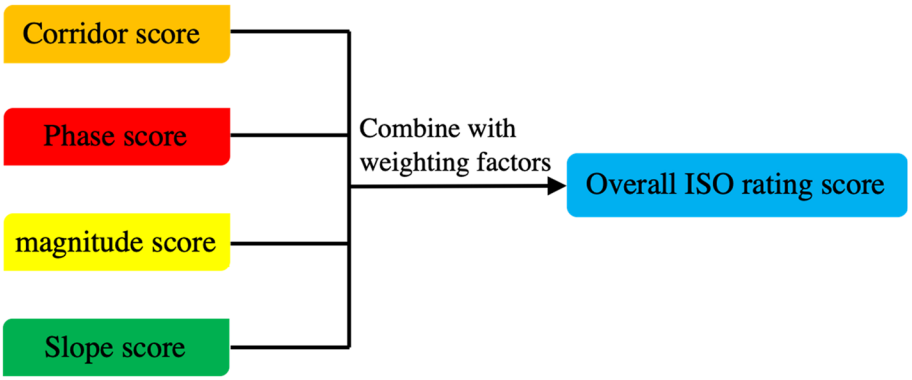


**Fig. 9** The structure of ISO validation metric

combination of the four scores considering each score's weight provides an overall objective rating that is a scalar number. Higher ISO score indicates stronger correlation between two different time series. For detailed computation of each score and overall ISO rating, can be found in Barbat et al. (2013) and International Organisation for Standardization (2014).

## 4 Case study: correlation of vehicle crashworthiness CAE data with test data

The proposed two approaches are applied to the fusion of CAE data with test data (as illustrated in Fig. 1), to improve the prediction accuracy of CAE model for new crash speed. Specifically, we focus on predicting crash pulse or deceleration since different impact speeds will produce different pulses. The main challenge is that vehicle deceleration/pulse is a highly nonlinear and complex output, involving many design variables and considerable uncertainty. As shown in Fig. 10, the predicted pulse/deceleration response exhibits highly nonlinear and non-stationary behavior over time under different crash speeds.

In addition to the highly nonlinear and non-stationary CAE response, there is noticeable discrepancy between the CAE prediction and the response of actual test as depicted in Fig. 11. It is therefore necessary to correlate the CAE model with test data to improve the CAE prediction accuracy.

In this study, the baseline FE model is the CAE model. The underlying true model is never known due to model form error. Thereby, the CAE model needs to be corrected using actual test data in order to accurately predict vehicle crash responses. To apply the proposed approaches, we first collect CAE data from a CAE simulation model. $N_e = 1009$ sets of speed, frontal and rear weights are first generated as training inputs using Latin hypercube sampling with uniform distribution [0, 35.2], [1922, 2185], and [1349, 1770], respectively. The simulation data is collected based on input samples using the CAE model. For each pulse, the output sequence is vehicle acceleration with data duration of 100 ms and a sampling frequency of 12.5 kHz. Thereby the time interval to measure acceleration is 0.08 ms at 12.5 kHz, which is commonly used in vehicle impact tests. The recorded test data are usually obtained at this frequency. In this study, actual tests and CAE model employ the same sampling frequency of 12.5 kHz to sample data for the sake of fair comparison. However, it is flexible in CAE model to use any other sampling frequency. From historical crash tests, only $N_c = 11$ sets of test data are available for different speed, weights, and configuration, as shown in Table 1, since crash tests are very expensive to perform. Remark that in this case study, all the CAE and test data have been

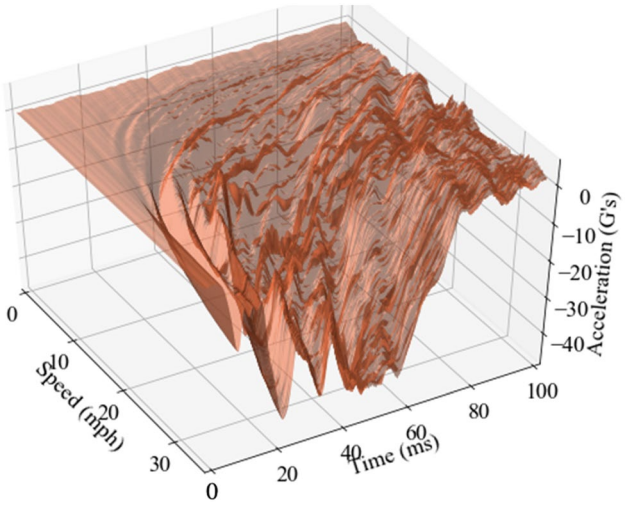


**Fig. 10** CAE prediction of deceleration for different crash speeds

preprocessed using SAE 60 Hz filter, which is a common practice for pulses from vehicle structural analysis.

### 4.1 Results of the time-domain approach

In the first place, we applied time domain-based approach for the fusion of CAE data and test data to improve the prediction accuracy of the CAE model. As discussed in Sect. 3.1, a low-fidelity TCN is first trained using the 1009 sets of CAE data, which is split into 90% for training and 10% for validation. The transfer learning method is then used to update the pre-trained low-fidelity TCN into a multi-fidelity TCN using a selected set of tests. More specifically, 10 out of the 11 tests given in Table 1 are selected for training the multi-fidelity TCN. The remaining one test is assumed to be unknown and is used to verify the accuracy of the final predictive model. For example, when we intend to predict the response of test No. 1, the data from rest of tests are used to train the multi-fidelity TCN. Table 2 summarizes the implementation of TCN and the user-defined hyperparameters. The TCN model was developed using the PyTorch library in Python. The input size is 4, including time, initial speed, vehicle frontal weight, and rear weight. The output vector is 1, e.g., acceleration at each time step. The number of channels $N_{ch}$ in TCN is designed as 600, the dilated casual convolution is fully connected with dilation factor $2^i$, $i = 0, 2, \ldots, N_{ch}$ for the $i$th layer, and a filter size of 20. A dropout rate of 0.05 is added for regularization after dilated convolution. The TCN model is assumed to contain one residual block. The Adam optimizer is used to train hyperparameters with a learning rate of 3e−4. The cosine annealing schedule is set as an optimization scheduler to recursively modify the learning rate with a maximum iteration of 100. The number of training epoch and prediction points are respectively

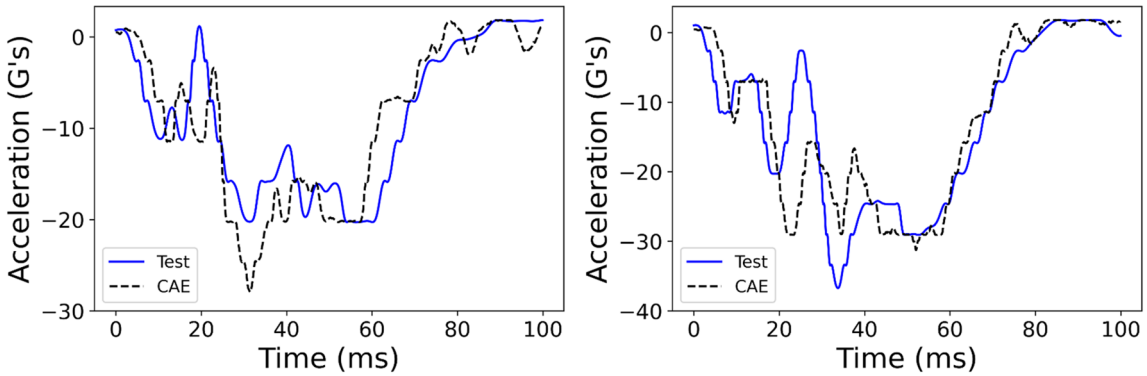


**Fig. 11** Comparison between CAE and test

**Table 1** Configuration of vehicle-to-rigid barrier full frontal test design

| Nos | Speed ($v_0$, mph) | Front weight ($w_f$, lbs) | Rear weight ($w_r$, lbs) | P/T configuration | FWD/AWD |
|---|---|---|---|---|---|
| 1 | 8.1 | 1994.67 | 1400 | Type I | AWD |
| 2 | 12.1 | 2163.74 | 1592.95 | Type II | AWD |
| 3 | 16.7 | 2120.14 | 1600.23 | Type I | FWD |
| 4 | 22 | 2237.39 | 1765.26 | Type II | AWD |
| 5 | 24.9 | 2100.43 | 1554.56 | Type I | FWD |
| 6 | 25.2 | 2155.42 | 1707.12 | Type II | AWD |
| 7 | 25.2 | 2058.97 | 1507.84 | Type I | FWD |
| 8 | 35.2 | 2179.31 | 1745.51 | Type II | AWD |
| 9 | 35.1 | 2107.68 | 1556.64 | Type I | FWD |
| 10 | 35.2 | 2166.85 | 1737.20 | Type II | AWD |
| 11 | 35.2 | 2168.93 | 1694.65 | Type II | AWD |

For case 1, the rear weight is intentionally set to 1400 as reference. All other weight data have been scaled accordingly

**Table 2** TCN parameter settings for time domain approach

| Input size | 4 | Dropout | 0.05 |
|---|---|---|---|
| Output size | 1 | No. of residual block | 1 |
| No. of channel | 600 | Optimizer | Adam |
| Filter size | 20 | Learning rate | 3.00E−04 |
| Dilation factor | $2^i, i = 0, 2, \cdots,$ 600 ($i$ is No. of layers) | Training epoch | 1000 |

10,000 and 1250 (sampling frequency 12.5 kHz × 100 ms). Note the training of LF_TCN and MF_TCN have the same design of the neural network. In this paper, prediction results of test Nos. 3–5, 7, 9, and 10 are presented for the sake of explanation. In addition, as shown in Fig. 11, the portion of acceleration data that increases with time is removed since we focus on deceleration (decrease in acceleration) before vehicle is bouncing back during a crash test.

As described in Sect. 3, the time-domain method builds the low-fidelity TCN (LF_TCN) using CAE data, and then adopts the transfer learning and test data to fine-tune LF-TCN into multi-fidelity TCN (MF_TCN). To show the predictive performance of LF_TCN and the necessity of using transfer learning and test data to fine-tune, we compare the results of LF_TCN with that of CAE and test data, as shown in Fig. 12. It is observed that accelerations predicted by LF_TCN (denoted as red dashed lines) have high agreements with those from CAE model (denoted as green dashed lines). This can be attributed to the fact that all LF_TCN models are trained using CAE data. Nevertheless, the noticeable gap between prediction by LF_TCN and test data still exists, especially for test Nos. 5, 7, and 9–10, suggesting the poor performance of using LF_TCN for predicting test data, as no test data is involved during LF_TCN training. It is worth mentioning that for tests Nos. 3–4, it appears that prediction by LF_TCN is closer to test data compared to prediction for other tests. Because these two tests have relatively small discrepancies between

CAE data and test data. Acceleration obtained from LF_TCN matches with CAE data, leading to more agreement between LF_TCN prediction and test data.

Figure 13 shows the comparison of results obtained from the actual test, MF_TCN, and the CAE model. It is observed that the CAE model gives the most diverged

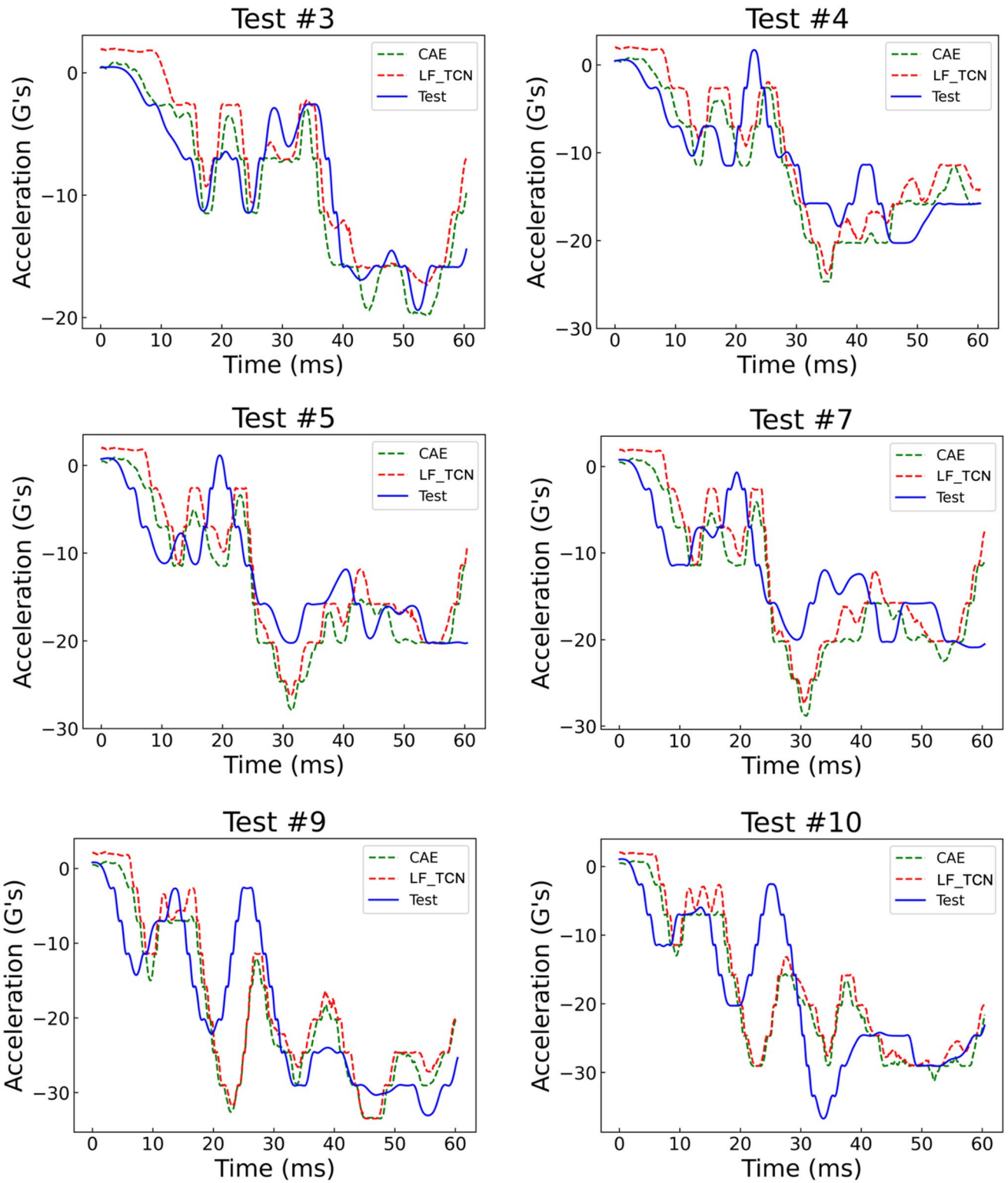


**Fig. 12** Comparison of results obtained using LF_TCN (i.e., time domain)

prediction from true responses because of the imperfection of the CAE model. The predicted curves (red dashed lines) by the MF_TCN model for test Nos. 3 and 4 are not good. For a certain time (e.g., acceleration prediction at 30–60 ms), it is even worse than that from the CAE model.

However, the MF_TCN prediction are very close to those from actual tests for test Nos. 5, 9, and 10. The difference between the predicted pulse curves and the actual tests are significantly reduced. This can be attributed to the fact that when predicting test Nos. 3 (16 mph) or 4 (22 mph), the rest of test data are used as training data which do not contain any information about crash test at speed 16 mph or 22 mph. Among the 11 tests given in Table 1, there is only one set of data at 16 mph and one set of data at 22 mph. However, the training data for predicting test Nos. 5, 7, and 10 have some similar information about the test data because the 11 sets of test data include multiple sets of tests at around 25 mph and 35 mph. Therefore, it is understandable that tests Nos. 5, 7, and 10 are predicted more accurately than tests Nos. 3 and 4.

Table 3 lists the ISO scores computed by LF_TCN and MF_TCN, respectively, to show the improvement by transfer learning and using test data. The Table also highlights the maximum ISO scores in the prediction period of a certain test, denoted by the bold values. As observed in Table 3, the MF_TCN results in higher ISO scores compared to those from LF_TCN for most test cases, illustrating the prediction accuracy is significantly enhanced by MF_TCN. However, for tests Nos. 3 and 4, LF_TCN gives higher overall ISO scores. As explained in Fig. 12, LF_TCN prediction matches with CAE data, and CAE data is closer to test data for tests Nos. 3–4, leading to more accordance between LF_TCN prediction and test data and thus higher ISO scores. On the other hand, one of the reasons that MF_TCN gives lower ISO scores for tests Nos. 3–4 is that when training MF_TCN model, the training data do not contain any information on these two tests (e.g., similar initial vehicle speed as to-be-predicted tests). Overall, the results indicate that the trained MF_TCN models can improve the prediction accuracy of the CAE model by fusing the CAE data with a small amount of test data.

### 4.2 Results of the displacement-domain approach

The displacement domain-based approach presented in Sect. 3.2 is also applied for the fusion of CAE data with test data. In this approach, the deceleration responses as a function of time in CAE and actual test are first converted to a function of displacement. The bijective projection is employed to realize one-to-one mapping between the 10 sets of test data for training and the corresponding CAE data. Note that the remaining one test data is used as validation data for prediction which is the same as that for the time-domain method. After that, a GPR model is built using training data to represent the relation between inputs and unmodeled physics of the nonlinear spring constant. The GPR model is built using scikit-learn library in Python and its hyperparameters is summarized in Table 4. The training inputs consist of 1500 length of 4 dimensions, e.g., initial speed, frontal, rear weight, and displacement resulting from 10 out of 11 test data. The training output include 1500 length of one dimension, that is, model bias calculated using ten test data and corresponding CAE data. The initial value of hyperparameters is set to 2. The hyperparameters range from 1e−5 to 2e3. Alpha is defined as 1e−5 to ensure a positive definite kernel matrix. The number of repeats of optimizer to find the optimal hyperparameters is 10. The kernel in designed GPR model is a combination of ConstantKernel and MaternKernel. The optimizer is selected as a default setting of L-BFGS-B algorithm. Specifically, given an initial speed, displacement and acceleration, bias in acceleration between test and CAE model at each time step is estimated. The displacement, acceleration, and velocity are finally predicted in an iterative manner using Eq. (28).

Note that the predicted displacements during the period of vehicle bouncing back are removed. It means that model prediction is performed excluding the time period when impact-induced displacement decreases (as illustrated in Fig. 14). Mathematically, corresponding gradient of displacement curve is less than zero. The core idea in the second approach is that the model is considered as a dynamic modeling problem which involves solving differential equations. The bias function is characterized by trained a GPR model and directly added to responses acquired from the CAE model. Model bias is therefore statistically compensated, resulting in accuracy improvement in model prediction. Figure 15 presents the displacements obtained by numerical integration and CAE simulation for test Nos. 3 and 9. The results from two methods exhibit high correlation, indicating the displacements computed by the numerical integration is reliable, and model prediction can be conducted by recursively solving differential equations.

Figure 16 shows the model bias of the CAE model prediction at different crash speeds, which is obtained using the actual test data using the projection method presented in Sect. 3.2. The red dashed circle highlights the most significant model bias at a certain displacement caused by an impact. It is not surprising that model bias tends to increase with speed, since higher crash speed will result in more impact energy and more complicated vehicle damage and deformation.

Figure 17 presents the comparison of the predicted pulses from the proposed method in the displacement domain and the CAE model. Since the GPR model enables for probabilistic prediction in the displacement domain,

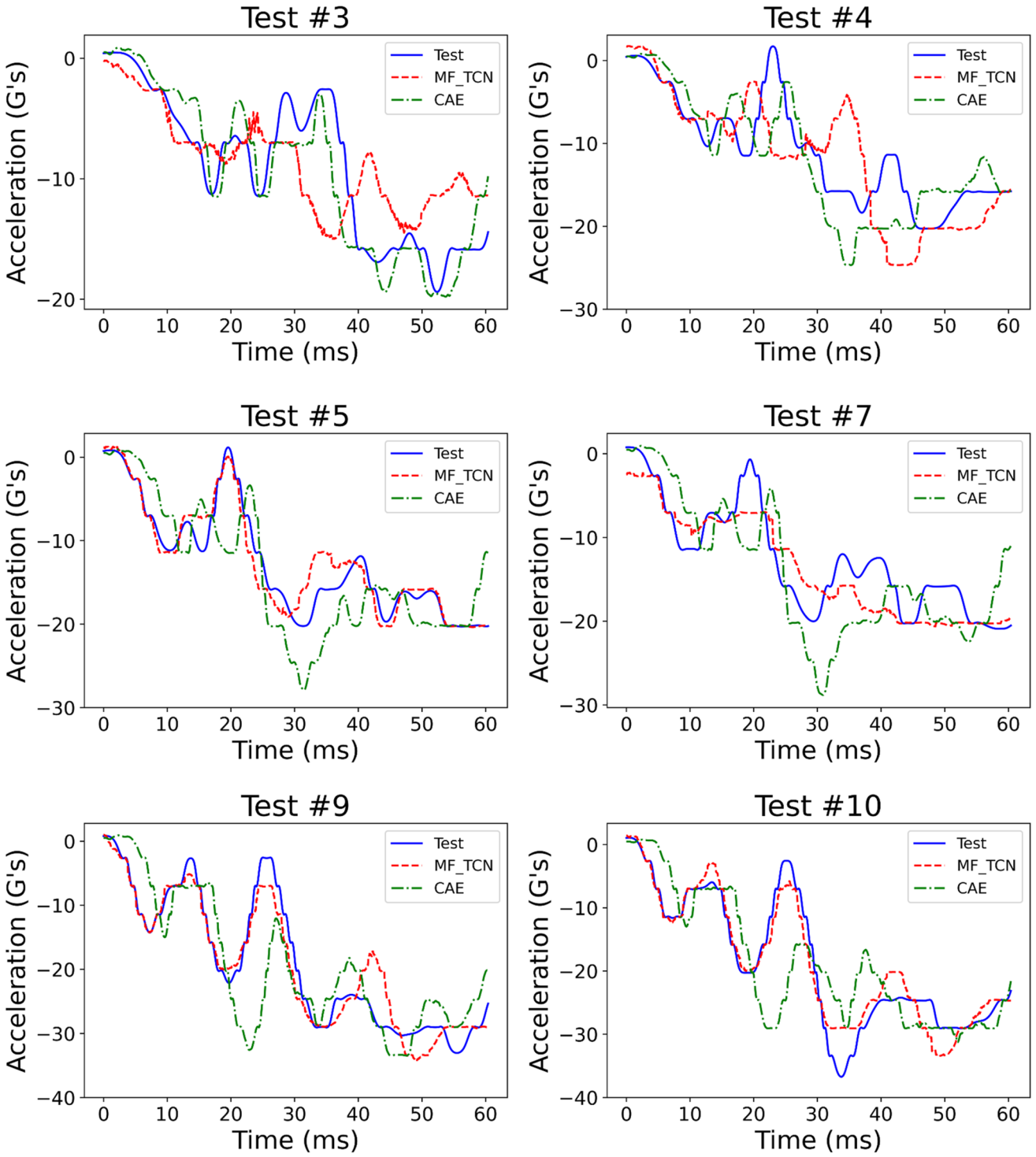


**Fig. 13** Comparison of results obtained using MF_TCN (i.e., time domain)

we also plotted the 95% confidence interval of the predictions in the figure. As shown in Fig. 17, the difference between pulse curves from the proposed method (i.e., denoted as "Mean prediction") and the actual test is much smaller than its counterpart between the CAE model and the actual test. It indicates that the proposed method effectively improves the prediction accuracy of the CAE model by correlating the CAE model with the test data in the displacement domain. Especially for Nos. 3, 5, and 10, the

**Table 3** Comparison of LF_TCN and MF_TCN using the ISO scores

| Test Nos | Time period | ISO score | | |
|---|---|---|---|---|
| | | (0, 20) ms | (0, 40) ms | (0, 60) ms |
| 3 | LF_TCN | 0.451 | **0.667** | **0.753** |
| | MF_TCN | **0.727** | 0.611 | 0.595 |
| 4 | LF_TCN | 0.406 | 0.581 | **0.616** |
| | MF_TCN | **0.692** | **0.584** | 0.420 |
| 5 | LF_TCN | 0.363 | 0.566 | 0.625 |
| | MF_TCN | **0.863** | **0.850** | **0.878** |
| 7 | LF_TCN | 0.365 | 0.527 | 0.579 |
| | MF_TCN | **0.691** | **0.762** | **0.785** |
| 9 | LF_TCN | 0.559 | 0.615 | 0.476 |
| | MF_TCN | **0.938** | **0.916** | **0.878** |
| 10 | LF_TCN | 0.526 | 0.566 | 0.487 |
| | MF_TCN | **0.963** | **0.917** | **0.903** |

**Table 4** GPR parameter settings for displacement domain approach

| Input size | $1500 \times 4$ | Alpha | $1e-5$ |
|---|---|---|---|
| Output size | $1500 \times 1$ | No. of training repeats | 10 |
| Initial parameter | 2 | Kernel | ConstantKernel × MaternKernel |
| Parameter bounds | (1e−5, 2e3) | Optimizer | Default: L-BFGS-B algorithm |

The details on ConstantKernel, MaternKernel and L-BFGS-B algorithm refer to scikit-learn library

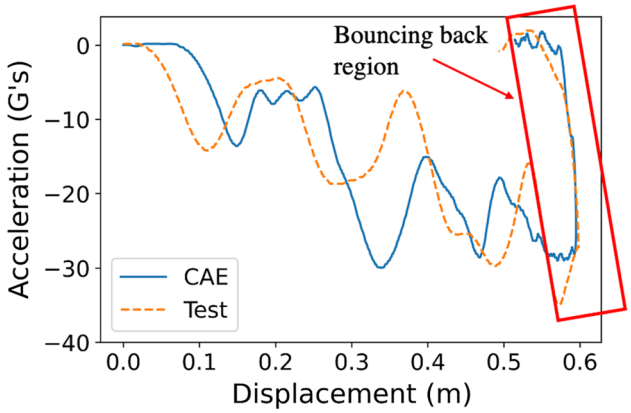


**Fig. 14** Removal of displacement bouncing back

predicted pulse curves from the proposed method are very close to that of the actual tests.

Moreover, unlike the time domain-based approach which is unable to accurately predict response of tests Nos. 3 and 4, the displacement domain-based approach

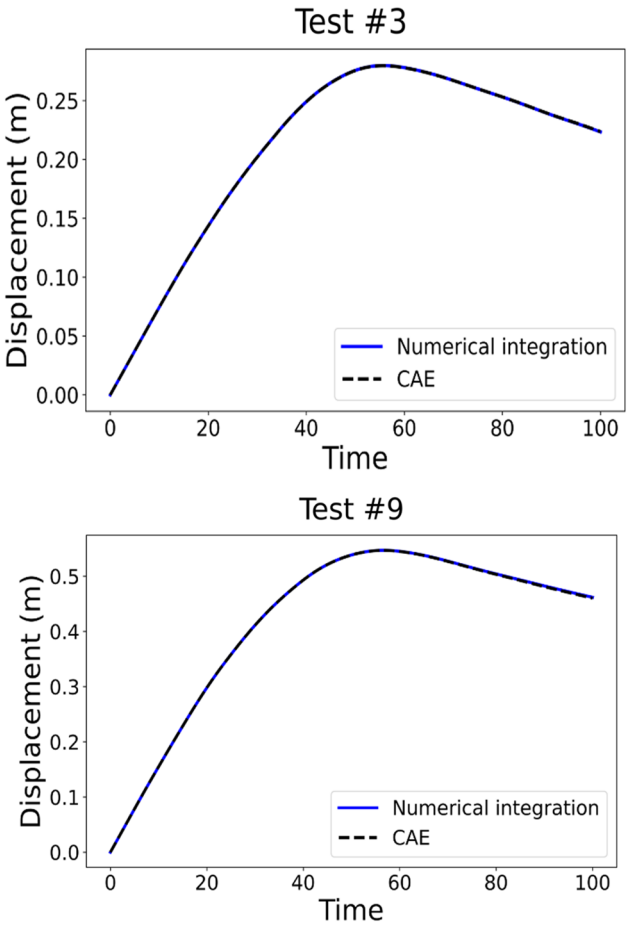


**Fig. 15** Displacement comparison between CAE and numerical integration

gives good accuracy in predicting these two sets of tests. A possible reason is that the displacement domain-based approach is physically more explainable and recovered the unmodeled physics of the CAE model using a ML correction term. In general, the results show that the proposed displacement domain-based approach clearly improves the prediction accuracy of the CAE model by correlating CAE data with test data in the displacement domain.

### 4.3 Quantitative comparison of different methods using the ISO score

The main objective of this research is to address the challenges of the discrepancy between CAE model prediction and the actual response of vehicle crash tests. Two ML-based approaches are proposed to correct the CAE model and improve predictive accuracy. It would be necessary to make a comparison between predicted accelerations and actual measured data in order to show whether the predictive performance is enhanced or not. The ISO metric is used to

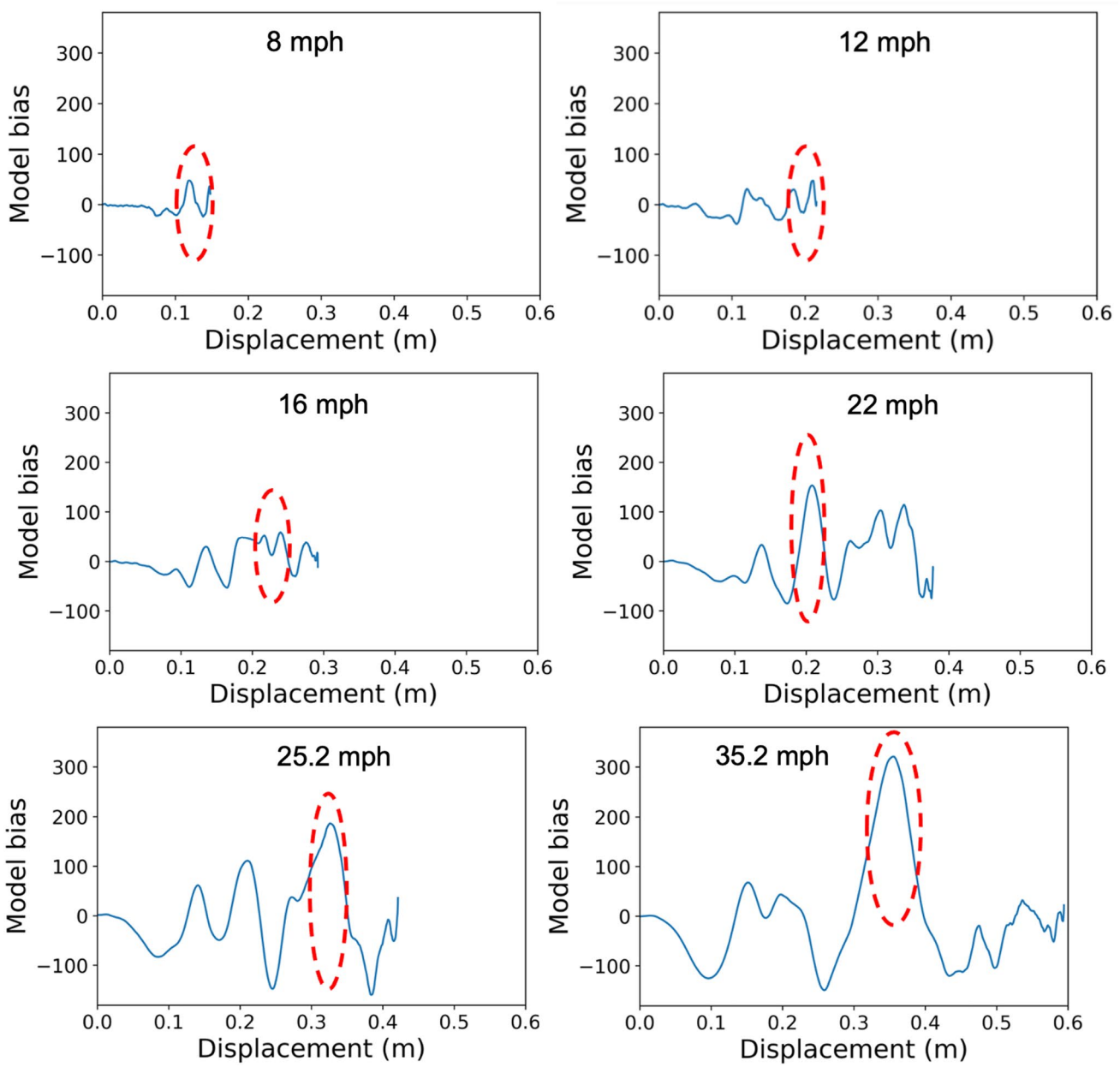


**Fig. 16** Model bias at different crash speeds

measure the degree of agreement between model prediction and real test data. To quantitatively compare different methods, the ISO metric presented in Sect. 3.3 is used to evaluate the prediction accuracy of the proposed time-domain and displacement-domain approaches. The higher ISO validation score is, the stronger correlation between prediction and test data is. In other words, a higher ISO score indicates a better performance in model prediction. Table 5 summarizes the overall ISO score of the proposed two approaches for three different time periods, namely 0–20 ms, 0–40 ms, and 0–60 ms. The bold values in the table represent the maxium ISO scores in a certain prediction period. It is shown that the overall ISO scores of the proposed two approaches for the three time periods are much higher than those of the CAE model. It means that the overall accuracy of the prediction is improved by fusing CAE data with test data using the proposed two approaches. In addition, it is observed that for test Nos. 3 and 4, the ISO scores of the time-domain approach are smaller than those of the displacement-domain. It indicates that more accurate prediction for test Nos. 3 and 4 can be obtained using the displacement-domain approach. For the prediction of the other tests (Nos. 5, 9, and 10), the time-domain approach gives higher ISO scores than the displacement-domain approach (except for test No. 7, both

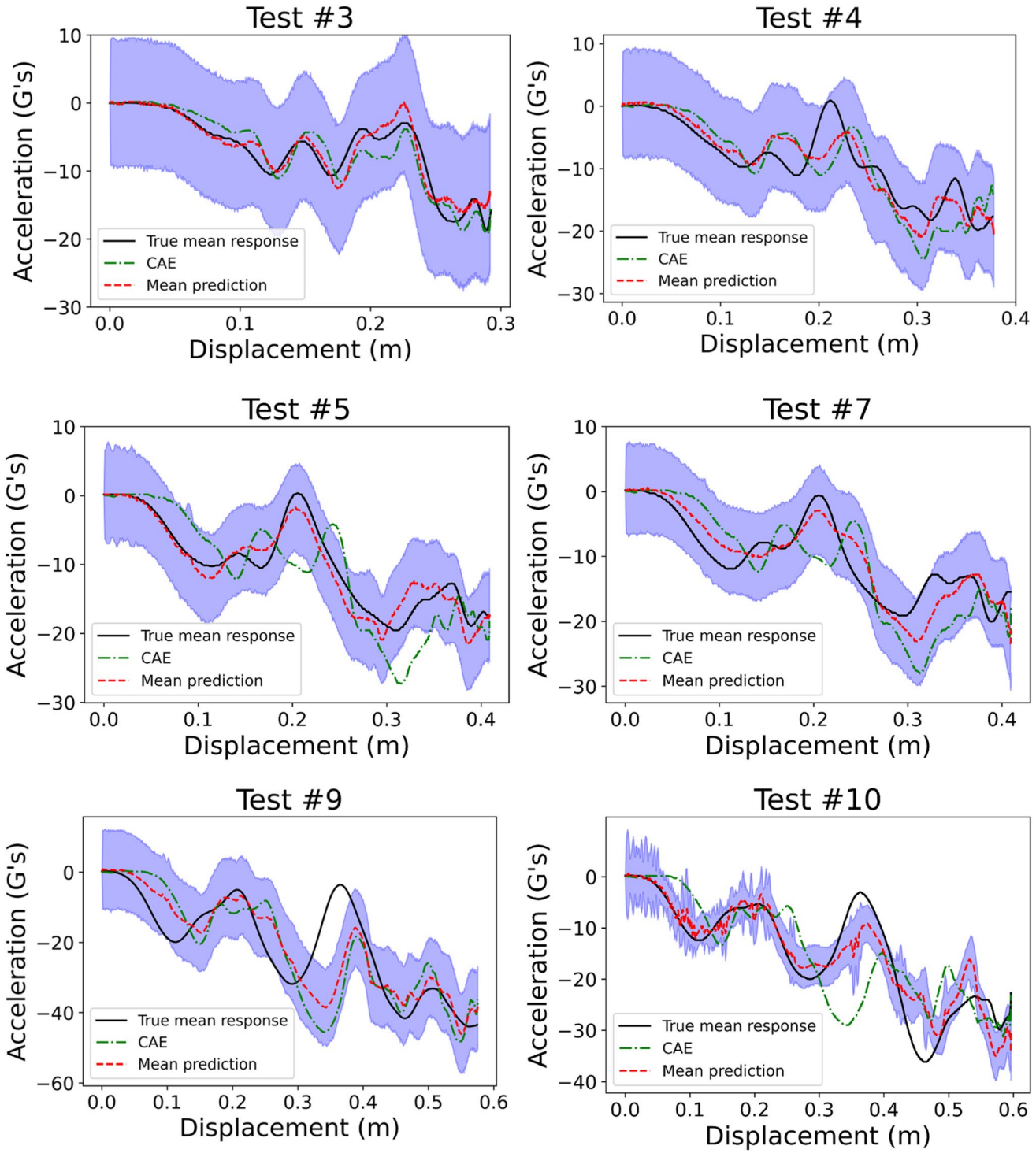


**Fig. 17** Result comparison in displacement domain

approaches have similar ISO scores). This implies that tests Nos. 5, 7, and 10 can be more accurately predicted by the time domain approach. It can be concluded from Table 3 that the predictive performance by the time-domain approach is sensitive to information included in the training data. The displacement-domain approach is more stable and robust to the configuration of the training data. More specifically, if training data contains very relevant information regarding prediction, such as predicting test Nos. 5, 9, and 10 in this study, the time-domain approach enables to give more

**Table 5** Comparison of validation results of proposed two approaches using the ISO scores

| Test Nos | Time period | ISO score | | |
|---|---|---|---|---|
| | | (0, 20) ms | (0, 40) ms | (0, 60) ms |
| 3 | Time domain | 0.727 | 0.611 | 0.595 |
| | Displacement domain | **0.863** | **0.848** | **0.862** |
| | CAE | 0.774 | 0.768 | 0.822 |
| 4 | Time domain | 0.692 | 0.584 | 0.420 |
| | Displacement domain | **0.770** | **0.767** | **0.739** |
| | CAE | 0.651 | 0.658 | 0.681 |
| 5 | Time domain | **0.863** | **0.850** | **0.878** |
| | Displacement domain | 0.791 | 0.831 | 0.799 |
| | CAE | 0.477 | 0.565 | 0.656 |
| 7 | Time domain | 0.691 | 0.762 | 0.785 |
| | Displacement domain | **0.733** | **0.792** | **0.805** |
| | CAE | 0.494 | 0.535 | 0.617 |
| 9 | Time domain | **0.938** | **0.916** | **0.878** |
| | Displacement domain | 0.734 | 0.731 | 0.766 |
| | CAE | 0.645 | 0.665 | 0.527 |
| 10 | Time domain | **0.963** | **0.917** | **0.903** |
| | Displacement domain | 0.958 | 0.907 | 0.895 |
| | CAE | 0.647 | 0.617 | 0.541 |

accurate predictions than the displacement-domain approach. If training data does not have similar crash speeds as that for prediction, such as predicting test Nos. 3 and 4, the displacement-domain approach is better and has higher credibility than the time-domain approach.

Based on findings in Table 5, some practical aspects can be recommended when applying proposed two ML-based model bias correction approaches for real-world vehicle crashworthiness design. The selection of two approaches depends on the configuration of training data on hand. For example, time domain approach is more suitable for those cases that a certain information concerning prediction is included in available training data, such as similar vehicle speed in training data as to-be-predicted or target vehicle crash test. Otherwise, time domain approach may exhibit biased predictive performance, as prediction results for test Nos. 3–4 in Table 5. On the other hand, displacement domain approach has more general applicability due to less sensitivity to configuration of training data. Even if the information about to-be-predicted crash test is out of the entire domain of training data, which is a more common situation in reality as we do not know or have very little knowledge for future crash tests to be predicted, displacement domain approach still gives fairly good prediction results. In this context, the displacement domain approach would be preferable as the first run to perform response prediction for crash test. The time domain approach can be implemented as a supplementary tool for a cross-validation.

It should be noted that for training TCN model in time domain approach and GPR model in displacement domain approach, appropriate hyperparameters are essential for the most desirable model performance and need to be optimized. However, to our knowledge, it is very difficult to design the best combination of hyperparameters that produce the best prediction results for ubiquitous application. This is a long-standing question because the selection of hyperparameters is case-dependent. In this paper, hyperparameters in TCN model and GP model is determined through a rule of thumb based on authors' experience and current literatures that investigated the effect of hyperparameters on model performance and gave recommendations on selecting those hyperparameters, such as Bai et al. (2018) and Farha and Gall (2019) for TCN model, and Schulz et al. (2018) and Kleijnen (2009) for GPR model.

In summary, validation results using the ISO metric demonstrate that the proposed two approaches can successfully correlating CAE data with test data to improve prediction accuracy of the CAE model and thereby facilitate virtual tests-based CBA.

## 5 Conclusion

In this study, two ML model-based approaches are proposed in the time-domain and the displacement-domain for the fusion of CAE data and test data to improve prediction accuracy of CAE vehicle crashworthiness simulation model. In the time-domain approach, a low-fidelity TCN model is first built using simulated data from a CAE model. Limited test data was used due to high cost of acquisition of test data. Therefore, transfer learning is adopted to transfer existed knowledge from the pre-trained low-fidelity model to a multi-fidelity TCN model using a small number of crash tests. In the displacement-domain approach, the model of vehicle crashworthiness is treated as a dynamic modeling problem. A bijective projection method is proposed for the CAE simulation data and actual test data to analyze the mismatch between the CAE model and the actual tests. A GPR model is then constructed to recover the missing physics of the CAE model based on the analysis. The resulting GPR model is finally integrated with the CAE model in a dynamic analysis scheme to predict crash pulse response under a new speed. The proposed approaches are demonstrated through an industrial application of vehicle crashworthiness case study. The results show that the proposed approaches can provide a better prediction compared to the original CAE model. Below is a summary of final conclusions:

1 The proposed two approaches have two completely different mechanisms. In the time domain, time is treated

as an input variable and the model becomes to be a static modeling problem. However, in the displacement domain, vehicle crash analysis is treated as a dynamic problem, which involves solving differential equations. The fusion of CAE data and test data is realized by adding a bias function to original CAE model.

2 The time-domain approach is limited to deterministic prediction. The displacement-domain approach, however, provides uncertainty about the model predictions. The probabilistic prediction is very useful for making risk-informed decision making under uncertainty. The displacement-domain approach in general performs better than the time-domain approach, even though both approaches are much better than the CAE model.

3 Based on ISO objective validation metric, it shows the performance of the two approaches is affected by the tests used for training. For example, the time-domain approach may show better prediction when the tests used for training includes similar test for prediction. While the displacement-domain approach may more accurately predict the pulse response, even if the training data does not contain tests with speed that is like that for prediction.

It is noted that current study does not apply the proposed approaches for CBA, and therefore there are no certification matrices in the CBA process. However, the proposed ML-based approaches exhibit potential capability of application for CBA process, e.g., reduce model error and significantly improve model prediction using very limited test data. The evaluation and CBA using proposed approaches is a worth research direction and will be left in the future study. Furthermore, although the proposed two approaches show good performances in this study, they may need to be further examined through different types of vehicles and other datasets in our future study.

**Funding** Funding for this work was provided by Ford Motor Company through the University Research Program. The support is gratefully acknowledged. The first author would also like to thank Mr. Xufeng Huang and Dr. Chen Jiang for the help of TCN code.

## Declarations

**Conflict of interest** The authors have no relevant financial or nonfinancial conflicts of interest to disclose.

**Replication of results** General information about vehicle specifications, test conditions, and pseudocode of the proposed method are provided in the manuscript. Detailed information related to vehicle specifications cannot be shared due to confidentiality. The other data and codes will be available upon request.

## References

Albawi S, Mohammed TA, Al-Zawi S (2017) Understanding of a convolutional neural network. In: 2017 International conference on engineering and technology (ICET), 2017. IEEE, pp 1–6

Arendt PD, Apley DW, Chen W (2012) Quantification of model uncertainty: Calibration, model discrepancy, and identifiability. J Mech Des 134(10):100908

Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. In: International conference on learning representations (ICLR) workshop, 2018, vol 1803, p 01271

Barbat S, Fu Y, Zhan Z, Yang R-J, Gehre C (2013) Objective rating metric for dynamic systems. In: Enhanced safety of vehicles, Seoul, Republic of Korea, vol 2, no. 3, 2013

Caputo F, Lamanna G, Perfetto D, Chiariello A, Di Caprio F, Di Palma L (2021) Experimental and numerical crashworthiness study of a full-scale composite fuselage section. AIAA J 59(2):700–718

Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555

Du X, Jiang B, Zhu F (2021) A new method for vehicle system safety design based on data mining with uncertainty modeling. Eng Struct 247:113184

Fang J, Sun G, Qiu N, Kim NH, Li Q (2017) On design optimization for structural crashworthiness and its state of the art. Struct Multidisc Optim 55(3):1091–1119

Farha YA, Gall J (2019) MS-TCN: multi-stage temporal convolutional network for action segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp 3575–3584

Gu X, Sun G, Li G, Mao L, Li Q (2013) A comparative study on multiobjective reliable and robust optimization for crashworthiness design of vehicle structure. Struct Multidisc Optim 48(3):669–684

Guida M, Manzoni A, Zuppardi A, Caputo F, Marulo F, De Luca A (2018) Development of a multibody system for crashworthiness certification of aircraft seat. Multibody Syst Dyn 44(2):191–221

Happian-Smith J (2001) An introduction to modern vehicle design. Elsevier, Oxford

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

Huang X, Xie T, Wang Z, Chen L, Zhou Q, Hu Z (2022) A transfer learning-based multi-fidelity point-cloud neural network approach for melt pool modeling in additive manufacturing. ASCE–ASME J Risk Uncertain Eng Syst B 8(1):011104

International Organisation for Standardization (2014) Road vehicles—objective rating metric for non-ambiguous signals, ISO/TS 18571:2014

Jackson KE, Fasanella EL, Lyle KH (2006) Crash certification by analysis—are we there yet? In: AHS International 62nd Annual Forum and technology display, 2006

Jiang C, Hu Z, Liu Y, Mourelatos ZP, Gorsich D, Jayakumar P (2020) A sequential calibration and validation framework for model uncertainty quantification and reduction. Comput Methods Appl Mech Eng 368:113172

Kennedy MC, O'Hagan A (2001) Bayesian calibration of computer models. J R Stat Soc B 63(3):425–464

Kleijnen JP (2009) Kriging metamodeling in simulation: a review. Eur J Oper Res 192(3):707–716

Kohar CP, Greve L, Eller TK, Connolly DS, Inal K (2021) A machine learning framework for accelerating the design process using CAE simulations: an application to finite element analysis in structural crashworthiness. Comput Methods Appl Mech Eng 385:114008

Lanzi L, Castelletti LML, Anghileri M (2004) Multi-objective optimisation of composite absorber shape under crashworthiness requirements. Compos Struct 65(3–4):433–441
Lea C, Vidal R, Reiter A, Hager GD (2016) Temporal convolutional networks: a unified approach to action segmentation. In: Computer vision—ECCV 2016 workshops, proceedings, Part III 14, 2016, Amsterdam, The Netherlands, 8–10 and 15–16 October 2016. Springer, pp 47–54.
Lea C, Flynn MD, Vidal R, Reiter A, Hager GD (2017) Temporal convolutional networks for action segmentation and detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp 156–165
Li L, Lin Q, Ming Z (2021) Multi-objective optimization using self-organizing decomposition and its application to crashworthiness design. Appl Soft Comput 101:107002
Olivares G, Acosta JF, Yadav V (2010) Certification by analysis I and II. In: The joint advanced materials and structures meeting, 2010
Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
Park J-S, Jeon J (2002) Estimation of input parameters in complex simulation using a Gaussian process metamodel. Probab Eng Mech 17(3):219–225
Quinonero-Candela J, Rasmussen CE (2005) A unifying view of sparse approximate Gaussian process regression. J Mach Learn Res 6:1939–1959
Rasmussen CE (2003) Gaussian processes in machine learning. In: Summer school on machine learning, 2003. Springer, pp 63–71
Rasmussen CE, Nickisch H (2010) Gaussian processes for machine learning (GPML) toolbox. J Mach Learn Res 11:3011–3015
Sarin H, Kokkolaras M, Hulbert G, Papalambros P, Barbat S, Yang R-J (2008) A comprehensive metric for comparing time histories in validation of simulation models with emphasis on vehicle safety applications. In: International design engineering technical conferences and computers and information in engineering conference, 2008, vol 43253, pp 1275–1286
Schulz E, Speekenbrink M, Krause A (2018) A tutorial on Gaussian process regression: modelling, exploring, and exploiting functions. J Math Psychol 85:1–16
Schwer LE (2007) An overview of the PTC 60/V&V 10: guide for verification and validation in computational solid mechanics. Eng Comput 23(4):245–252
Sinha K, Krishnan R, Raghavendra D (2007) Multi-objective robust optimisation for crashworthiness during side impact. Int J Veh Des 43(1–4):116–135
Sobester A, Forrester A, Keane A (2008) Engineering design via surrogate modelling: a practical guide. Wiley, Chichester
Sun G, Pang T, Fang J, Li G, Li Q (2017) Parameterization of criss-cross configurations for multiobjective crashworthiness optimization. Int J Mech Sci 124:145–157
Tang Z et al (2017) Data-driven train set crash dynamics simulation. Veh Syst Dyn 55(2):149–167
Thelen A et al (2022) A comprehensive review of digital twin—part 1: modeling and twinning enabling technologies. Struct Multidisc Optim 65(12):354
Thelen A et al (2023) A comprehensive review of digital twin—part 2: roles of uncertainty quantification and optimization, a battery digital twin, and perspectives. Struct Multidisc Optim 66(1):1
Wan R, Mei S, Wang J, Liu M, Yang F (2019) Multivariate temporal convolutional network: a deep neural networks approach for multivariate time series forecasting. Electronics 8(8):876
Wang X, Shi L (2014) A new metamodel method using Gaussian process based bias function for vehicle crashworthiness design. Int J Crashworthiness 19(3):311–321
Xi Z, Hao P, Fu Y, Yang R-J (2014) A copula-based approach for model bias characterization. SAE Int J Passeng Cars Mech Syst 7(2):781–786
Yin H, Wen G, Hou S, Chen K (2011) Crushing analysis and multiobjective crashworthiness optimization of honeycomb-filled single and bitubular polygonal tubes. Mater Des 32(8–9):4449–4460
Zeng J, Kim YH (2020) Identification of structural stiffness and mass using Bayesian model updating approach with known added mass: numerical investigation. Int J Struct Stab Dyn 20(11):2050123
Zeng J, Kim YH (2022) Probabilistic damage detection and identification of coupled structural parameters using Bayesian model updating with added mass. J Sound Vib 539:117275
Zeng J, Todd MD, Hu Z (2022) Probabilistic damage detection using a new likelihood-free Bayesian inference method. J Civ Struct Health Monit. https://doi.org/10.1007/s13349-022-00638-5
Zhan Z, Fu Y, Yang R-J (2013) On stochastic model interpolation and extrapolation methods for vehicle design. SAE Int J Mater Manuf 6(3):517–531
Zhan Z, Fu Y, Yang R-J (2014) A stochastic bias corrected response surface method and its application to reliability-based design optimization. SAE Int J Mater Manuf 7(2):262–268
Zhao Y, Jiang C, Vega MA, Todd MD, Hu Z (2022) Surrogate modeling of nonlinear dynamic systems: a comparative study. J Comput Inf Sci Eng 23(1):011001