Transportation Research Part C 126 (2021) 103009

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

A deep neural network inverse solution to recover pre-crash impact data of car collisions☆

Qijun Chen [a], Yuxi Xie [a], Yu Ao [a,b], Tiange Li [a], Guorong Chen [a,c], Shaofei Ren [a,b], Chao Wang [a], Shaofan Li [a,*]

[a] Department of Civil and Environmental Engineering, University of California, Berkeley, CA 94720, USA
[b] College of Shipbuilding Engineering, Harbin Engineering University, Harbin 150001, China
[c] School of Civil Engineering, Central South University, Changsha, Hunan 410075, China

ARTICLE INFO

Keywords:
Artificial intelligence
Car collision
Crashworthiness
Structural forensic analysis
Inverse solution
Machine learning
Traffic accident

ABSTRACT

In this work, we have successfully developed a data-driven artificial intelligence (AI) inverse problem solution for traffic collision reconstruction. In specific, we have developed and implemented a machine learning computational algorithm and built a deep neural network to determine and identify the initial impact conditions of car crash based on its final material damage state and permanently deformed structure configuration (wreckage).

In this work, we have demonstrated that the developed machine learning algorithm as an inverse problem solver can accurately identify initial collision conditions in an inverse manner, which are practically unique if we use permanent plastic deformation as the forensic data signatures. In other words, we think that the massive plastic energy dissipation process and the related big data will make final structure damage state insensitive to the initial car collision conditions. Thus, it provides an inverse solution for car crash forensic analysis by reconstructing the initial failure load parameters and conditions based on the permanent plastic deformation distribution of cars. This approach has general significance in solving the inverse problem for engineering failure analysis and vehicle crashworthiness analysis, which provides a key contribution for the unmanned autonomous vehicle and the related technology.

1. Introduction

The fatality and injury loss and the cost from structure damages caused by traffic accidents not only have significant impacts on the society but also seem to be unavoidable (Chong et al., 2005). As the age of the self-driving car and unmanned autonomous transportation is in the horizon, in recent years, researchers have actively developed artificial intelligence technology to determining factors that may significantly affect severity of driver injuries caused by traffic accidents, e.g. (Sohn and Lee, 2003; Chong et al., 2005; Bohn et al., 2013).

Piercing together what happened after a car crash used to be almost impossible, but now, thanks to advances in technology, this situation has been much improved. Some new vehicles are equipped with onboard video camera and crash data recorder (CDR), which can collect crash data directly from the crashed vehicle involved providing critical pre-crash parameters, such as vehicle speed, brake

☆ This article belongs to the Virtual Special Issue on IG005572 - VSI:Machine learning.

* Corresponding author.
E-mail address: shaofan@berkeley.edu (S. Li).

https://doi.org/10.1016/j.trc.2021.103009
Received 7 June 2020; Received in revised form 30 December 2020; Accepted 1 February 2021
Available online 10 April 2021

status, throttle position, ignition cycles, and seat belt status, leading up to as well as during the crash.

However, most car collisions happen very fast, and are often violent with many random factors, such as the tangential impact velocity that cannot be measured by CDR, the center of impact location, dynamic load condition, the precise road conditions or the conditions of road-tire interaction. Therefore, when a car crash accident happened, it often confuses the people involved, because some critical data and information are simply difficult to record, such as the relative contact velocity, the instantaneous offset position of the impact center, the instantaneous contact angle of the two cars, and the precise road condition at the accident site, which is independent from the brake status, among others. Since the pre-impact or pre-crash parameters determines the liability, to accurately find the pre-crash data posts a great challenge in car crash forensics, especially some close cases, while the involved parties always have their own versions of the story.

The gold standard in the traffic accident investigation is: *the pre-crash data determines the liability*, however, the data that are available in car crash forensic analysis are usually collected after the accident, and they mainly come from the information of the final state of crashed cars or the wreckage (see Fig. 1). The goal of the forensic analysis is to use post-crash data to find the pre-crash data, which is a typical and nevertheless challenging *inverse problem*.

There are a number of ways in which forensic investigators review and evaluate the causes and circumstances of a car accident. In the event of a complicated accident involving a serious injury or death, a specialized reconstructionist may be needed to analyze the crash site and to identify the cause of the collision. Forensic engineers often use post-crash evidences collected at the scene to put together a time-sequence of the car accident under investigation. After the reconstruction analysis is completed, forensic engineers usually use forensic simulation to show the collision sequence in a video format so that non-technical layman, such as juries, can easily understand the event. To be physically accurate, a simulation needs to be created based on the law of physics and pre-crash data or initial and boundary conditions. However, often times, this is not the case, and in fact many car accident simulations are not only inaccurate, but also based on misinformation. An often quoted reason for such inability to find precise pre-crash data is attributed to the non-uniqueness of the inverse problem solution, as well as our inability to understand the complex physical event of the crash accident in terms of physical modeling.

In some recent studies, the present authors have developed an artificial intelligence (AI) machine learning approach to use permanent plastic deformation to inversely find loading conditions or impact conditions of beam (Ren et al., 2018) and shell structures (Chen et al., 2019). In this work, we adopt this AI or data-driven approach to reconstruct of a car crash accident based on the post-crash data. The benefits of this data analytic approach are its accuracy, speed, and efficiency. In particular, we have developed an artificial neural network (ANN) computational algorithm, or a deep neural network to identify the pre-crash data or pre-crash impact conditions during traffic accidents (see Fig. 2).

Deep Learning is a very powerful AI method, and it has been applied to solve many engineering problems. The artificial neural network not only can be trained by experimental data, but also by the data from virtual reality (Jain et al., 1996). Virtual product development based on modeling and simulation is nowadays an essential tool in automobile industry. It has extensively used in car design to analyze the influence of design parameters on the weight, costs, functional properties, etc. of new car models (Haug et al., 1986). In fact, automobile mechanical engineers spend a considerable amount of their time analyzing these influences by inspecting the arising simulations one at a time. There have been researches focusing on using machine learning method to semi-automatically analyzing the data collected from finite element modeling and thereby significantly improving and assisting the overall engineering process e.g. (Bohn et al., 2013; Chong et al., 2005). Because of these reasons, an ANN model developed in this work is first illustrated by using virtual FEM data to demonstrate the validity and capacity of the proposed method.

In a broader sense, all of the engineering materials, products and structures are designed, manufactured or constructed with an

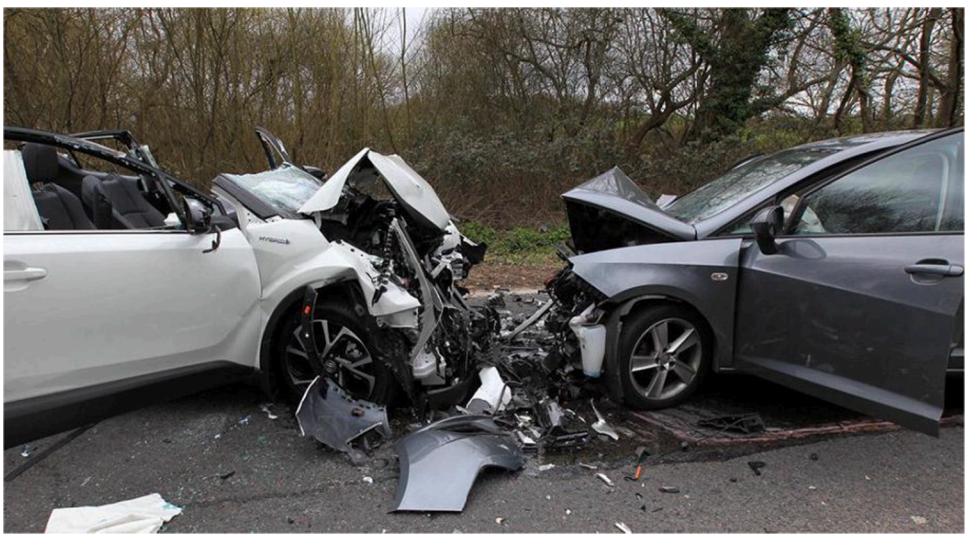

Fig. 1. Final damage state of a head-on car crash and its wreckage (from Steps To Follow After A Car Collision (2018)).

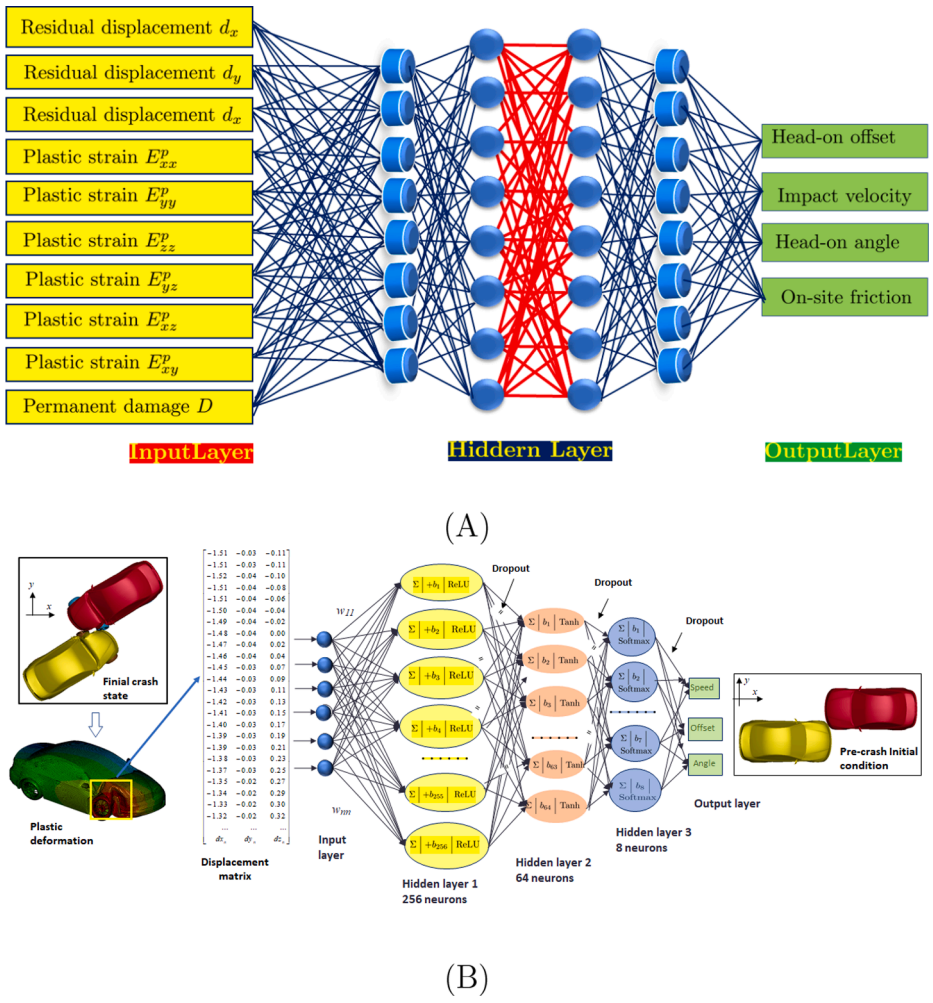

Fig. 2. (A) Concept of artificial neural network (ANN) for inverse solution of traffic collision, and (B) A detailed deep neural network (DNN) for identifying three main pre-crash data.

intention to function properly. However, they can fail, suffer damages or may not operate or function as intended due to various reasons including material or design flaws, extreme loading, etc. It is important to identify the causes of these failures or damage in order to improve the designs and detect any flaws in the materials or designs. One of the essential requirements of identifying the reasons of these failures is to know the loading conditions that lead to the failures.

Unfortunately, these loading conditions are not readily known while the forensic signatures such as the plastic strains or plastic deformations are easily measurable. For example, in car crashes, the impact loads on cars are not known, while the permanent deformations can be quantified after the crash. If the impact loads can be determined, it could potentially help insurance companies determine which party is responsible for the accident, and help car manufactures develop more realistic crashworthiness technologies.

What emerges from these considerations is an inverse problem of finding loading conditions from mechanical responses. This represents a reverse and disruptive approach of current engineering practice, in which the typical setup is to develop finite element models of structures, subject them to static and dynamic loading conditions, and then compute the resulting strains and residual displacements.

Machine learning and artificial intelligence encompass powerful tools for extracting complicated relationship between input and output sampling data, potentially through a training process, and then using the uncovered relationship to make predictions (Hastie et al., 2009; Nasrabadi, 2007). Machine learning and artificial intelligence have found a large number of successful applications in various fields beyond their birthplace in computer science (Sebastiani, 2002; Bratko et al., 2006; Sajda, 2006; ASCE, 2000). In recent years, there have been a number of studies devoted to applying machine learning techniques to conduct forensic materials analysis (Jones et al., 2018; Mena, 2016).

In this work, our goal is to develop a novel machine (deep) learning computational framework to solve the inverse problem by determining and identifying damage loading parameters (conditions) for structures and materials based on the engineering responses such as permanent or residual plastic deformation distribution or damage state of the structure.

This work combines the state of the art finite element modeling with recent advances in machine learning methodologies. This approach will advance the state of the art in forensic materials engineering, which seeks to examine material evidence and determine the original causes (Lei et al., 2019; Zheng et al., 2018; Zhou et al., 2018; Kirchdoerfer and Ortiz, 2018). We believe that machine learning based approaches can solve many previously intractable problems, with prior approaches incurring impractical computational costs due to the scale of the finite element models, the large degrees of freedom, and the complex and dynamic nature of the loading forces (Bengio et al., 2007).

We begin with an outline on the gathering process of the required data to train the machine learning algorithms. This will be followed by examples that demonstrate how we solve the inverse problem. We seek to demonstrate with these examples that the machine learning algorithms can accurately identify both static loading and impact loading conditions based on observed residual plastic strain, deformation or other features. A detailed description of the machine learning neural network algorithm will be presented in the later sections.

The paper is organized into six sections. In Section 2, we first discuss data collection, and followed by discussing the developed deep neural network (DNN) in Section 3. In Section 4. we provide all the material information used in the simulations, and in Section 5. we summarized all the simulation and validation results. Last in Section 6, we conclude the study with a few remarks.

2. Data collection

2.1. Finite element model and boundary conditions

Based on the geometry of a real BMW sedan, a 3D car model was generated and meshed by use of Solid3D. However, it is noticeable that for a relatively large scale simulation of plasticity and fracture are extremely time consuming. The geometry information of the car is shown in Table 1. The three-dimensional FEM car model is shown in Fig. 3(a). A car model contains three main parts which are the car body shell, the axels and the wheels. Then two car models were generated and positioned head-on to mimic the crashing position, as shown in Fig. 3(b).

The geometric model is input to the professional car crash simulation software LS-DYNA. All of the simulations in this study were carried out by used of LS-DYNA (Hallquist, 2007). It is a general-purpose, multi-physics nonlinear finite element program that is capable of simulating complex real-world problems. It has been extensively used by the automobile, aerospace, construction, military, manufacturing, and bioengineering industries. Since the thickness of car body shell is much smaller than the dimension of the car, the quadrilateral and triangle Belytschko-Tsay shell element (Eq. (2)) are adopted to model the car body, and a shell element with relative large thickness is used for to model the wheel. It may be noted that the Belytschko-Tsay shell element (Eq. (2)) is not a membrane element but a shell element, which has been extensively used in LS-DYNA for crashworthiness modeling and simulation of vehicles. The car axles, which are two solid components, were modeled by using eight-node linear solid elements. The influence of different mesh sizes was investigated in a preliminary study. Finally, an element size of around 2 cm was chosen in the FEM model for considerations of accuracy and computation cost, which leads to a total 39194 elements and 39919 nodal points in one car. Since the rotation of the wheel and the axles has less effect on the crash results, rigid connections between the wheels and the axles and between the axles and the car body were adopted. A nodal point in the button of each four wheels was fixed on vertical direction but free on horizontal directions. Thus, the cars can move and rotate in the X-Z plane but not on the Y direction. Except those four nodal points, the degree of freedom of all other nodal points have no constrains. Thus, the nodal points can deform on any direction. The global coordinate origin was set in the center point between two cars. The contact relationship between two cars was modeled by the Automatic_Surface_To_Surface module of LS-DYNA, and it is a recommended contact type for crash simulations in LS-DYNA, and the automatic contacts check for penetration on either side of a shell element, because the orientation of parts relative to each other cannot always be anticipated as the model undergoes large deformations. We listed all the main LS-DYNA simulation parameters in Table 2.

The static and dynamic coefficients of friction between all parts were set to 0.15. Hourglass energy, Stonewall energy, sliding interface and Rayleigh energy dissipations were computed and included in the energy balance. We choose the Johnson–Cook model as the material model for car structure (Johnson and Cook, 1985), which will be discussed in details in Section 4.

2.2. Data standardization

Most data-driven approach needs preprocessing the data that the user attempts to learn from. Often, raw data is comprised of attributes with varying scales. Although not required, we can get a boost in computational performance by carefully rescaling data, when the data is statistical.

Table 1
Geometry information of the model.

Features	Values
Length	4.082 m
Width	1.783 m
Height	1.192 m
Wheel base	2.492 m
Car body shell thickness	3.5 mm

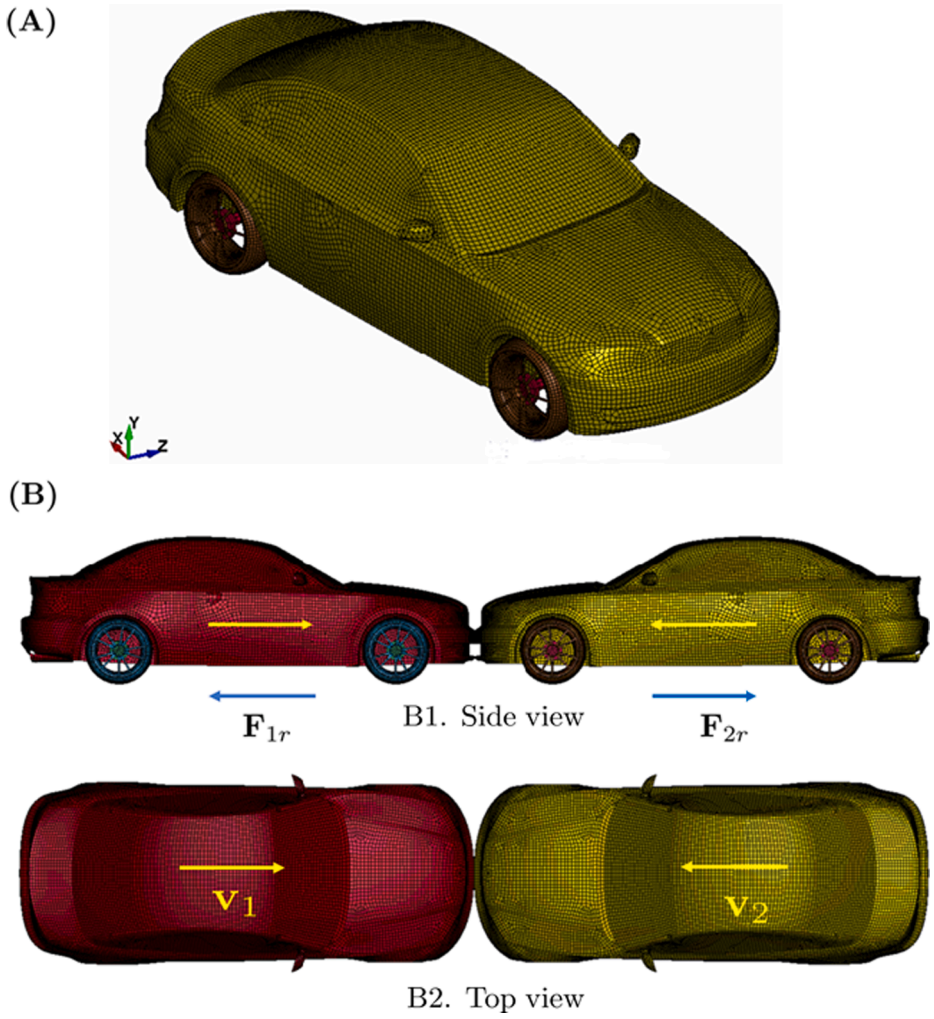

Fig. 3. Finite element passage car model and head-on impact load conditions.

Table 2
LS-DYNA modeling and simulation parameters.

Time step	1.0×10^{-4} s
Time integration algorithm	Newmark-β method ($\gamma = 0.5, \beta = 0$)
Shell element type	Belytschko-Tsay shell element (Eq. (2))
Thickness of the shell	3.5 mm (in car body)
Thickness of the shell	1 cm (in wheel)
Solid element type	Eight node brick element
Element number	39194
Element size	$\sim 2.0 \times 10^{-2}$ m
Contact type	CONTACT_AUTOMATIC_SURFACE_TO_SURFACE (a3)
Contact friction coefficient	0.15
Hourglass control type	3 (Flanagan-Belytschko)
Hourglass coefficient	0.1

The Z-score normalization will render a random data-set having a standard normal distribution with $\mu = 0$ and $\sigma = 1$, where μ is the mean (average) and σ is the standard deviation from the mean. Standard scores (also called z scores) of the samples are calculated as follows,

$$z = \frac{x - \mu}{\sigma}. \tag{1}$$

It may be note that data standardization is also a general requirement for many machine learning algorithms, such as principal component analysis (PCA) and neural network (O'Boyle et al., 2011; Greenwood and Williamson, 1966), because standardization is important in PCA. Since PCA is a variance maximizing approach Data standardization forces several different features into same range and standard deviation so that they are comparable regardless of possible different ranges of features and the influence of different units. In this study, different features have several different ranges, such as $[0, 10^{-6}]$ and $[0, 1]$. That is why standardization is necessary.

2.3. Pre-crash data recovery algorithm

Applying machine learning method for pre-crash data recovery is not a simple application of Tensorflow or any other common machine learning algorithms or methods. We found that it is an expert domain-specific knowledge dependent problem, and it not very accurate to directly apply the popular machine learning algorithms for multiple pre-crash data prediction.

Therefore, in this work, we have developed a three-neural network (3NN) algorithm to recover the pre-crash data, which contains three deep neural networks to predict the key pre-crash parameters in the following five-step procedure (see Fig. 6),

1. Building a neural network and use TensorFlow to predict the offset between two cars based on all the training data.
2. Based on the predicted offset, obtaining the new sets of training data from the whole training data using data filtering, and then building the second neural network.
3. Using the second neural network to predict the magnitude of the angles of two cars.
4. Based on the predicted offset and angles, using data filtering again to obtain the new sets of training data from the filtered training data in Step 2, and then building the third neural network.
5. Using the third neural network to predict the magnitude of the velocities of two cars.

The above algorithm is illustrated in Fig. 6. In the above pre-crash data recovery operation, we have four input variables: the permanent displacements in x, y, z directions, and the effective plastic strain in each element. The range of the input variables depend on pre-crash collision conditions. When the relative impact velocity is below 50 km/h, the range for the permanent displacements are (unit:meter): x-direction $(-1.7 \sim 0.3)$; y-direction $(-0.05 \sim 0)$; z-direction $(-0.11 \sim 0.49)$, and the range for the von-Mises plastic strain is $(0 \sim 0.94)$. The output of the above pre-crash data recovery algorithm are: impact location offset, angle, and relative impact velocity. For the training data used in our calculations, the angle of the output data are: offset (0.0 m $\sim$ 1.5 m); angle ($0^\circ \sim 45^\circ$), and the relative velocity: (10 km/h $\sim$ 70 km/h). From Table 3, one may find that there are total 34 cases of different car collision velocity scenarios, i.e. the red and blue color velocity pairs in Table 3. We choose four different discrete impact collision angles, i.e. $0^\circ, 15^\circ, 30^\circ$ and 45° and seven discrete offset distances, i.e. 0.0 m, 0.25 m, 0.5 m, 0.75 m, 1.0 m, 1.25 m and 1.5 m. Therefore, there are total $34 \times 4 \times 7 = 952$ collision cases in the training data set. The input data size at the training stage is: $932 \times 79112 \times 4$, and the input data size for the testing case is: 79112×4. During prediction stage, because there are many zeros in the original two car crash input data array depending on the collision severity, we can eliminate those zero entries by filtering the input data array. For example, for the collision case of two car velocity pair (35 km/h −15 km/h), the actual input data size reduces to 32683.

In the previous example, by identifying outlier values or error data or noise data and removing them based on threshold criterion of the previous results, we can further filter out some data, reduce the size of the testing input data for the second neural network to 32475. The input data set for the second neural network consists of these filtered data with the additional 3 offset prediction data; we can further reduces the size of the testing input data to 32366 by filtering out outliers and noise data to 32366. By combining these data with the predicted offset data and angle data, we have the testing input data for the third neural network. After the third neural network, finally we have the final predicted values for the pre-crash offset, angle, and relative impact velocity. It may be noted that in this work we use the standard feed-forward network. On the other hand, we also use back propagation (De Villiers and Barnard, 1993) to optimize all weights.

A key technical step in the above algorithm is data filtering, which is a process of choosing a smaller but essential part of data set to train the neural network, and it provides an efficiency process (Haining, 1993). For each step in Fig. 6, we represent in a different color,

Table 3
Impact collision speeds for crash car pair (km/h).

Car 1 / Car 2	10 km/h	20 km/h	30 km/h	40 km/h	50 km/h	60 km/h	70 km/h
0 km/h	10-0	20-0	30-0	40-0	50-0	60-0	70-0
10 km/h	10-10	20-10	30-10	40-10	50-10	60-10	70-10
20 km/h		20-20	30-20	40-20	50-20	60-20	70-20
30 km/h			30-30	40-30	50-30	60-30	70-30
40 km/h				40-40	50-40	60-40	70-40
50 km/h					50-50	60-50	70-50
60 km/h						60-60	70-60
70 km/h							70-70

Fig. 9 shows the flowchart of the inside algorithm for each step. Moreover, Fig. 10 reveals the Python code stricture of one neuron. By bringing in the above three neural networks together, the accuracy and speed of the computation will improve significantly, which will be discussed in detail in the following sections.

3. DNN algorithms

In this section, we discuss the developed deep neural network algorithm for the structural failure inverse problem solution.

3.1. Activation function

In artificial neural networks, the activation function of a node defines the output of that node, or "neuron", given an input or a set of inputs. This output is then used as input for the next node and so on until a desired solution to the original problem is found. It maps the resulting values into the desired range such as between 0 to 1 or −1 to 1 etc., depending upon the choice of activation function. For example, the use of the logistic activation function would map all inputs in the real number domain into the range of 0 to 1 (Specht, 1991; ASCE, 2000; Specht, 1990). A standard computer chip circuit can be viewed as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input. This is similar to the behavior of the linear perception in neural networks. However, only nonlinear activation functions allow such networks to compute nontrivial problems using only a small number of nodes (König et al., 2011). In the car collision study herein, we use five different activation functions: Rectified Linear Unit (ReLU) function, sigmoid function, tanh, softmax and ReLU square. They are expressed as follows,

$$f(x) = \begin{cases} 0, x < 0 \\ x, x \geqslant 0 \end{cases} \tag{2}$$

the sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

tanh function

$$f(x) = tanh(x), \tag{4}$$

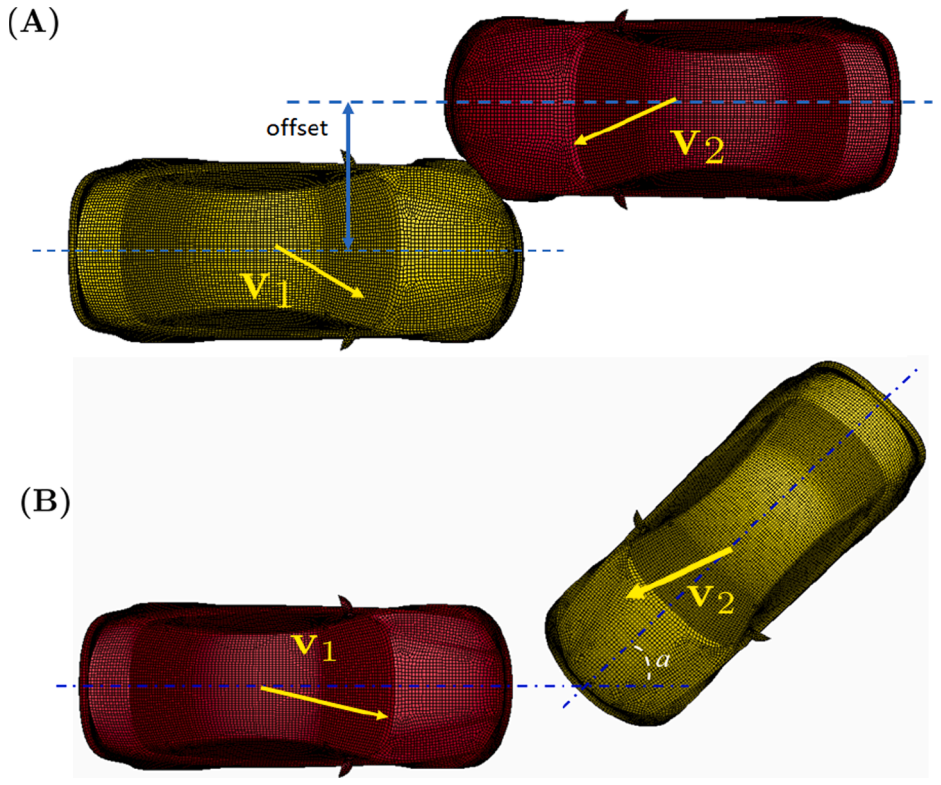

Fig. 4. Key pre-crash/impact conditions: (A) Horizontal offset distance from the center in the initial contact, and (B) Impact contact with oblique contact angle.

the softmax function,

$$f(x) = \frac{e^{x_i}}{\sum_{i=0}^{k} e^{x_i}} \tag{5}$$

and the ReLU square

$$f(x) = \begin{cases} 0, x < 0 \\ x^2, x \geqslant 0 \end{cases} \tag{6}$$

3.2. Training the deep neural network

After collecting training data, both the post-crash data as well as the pre-crash data, we start to design and train a deep neural network for predicting pre-crash parameters based on input post-crash data that is not in the trained data set. To some extent, the variety, quantity and quality of the data determine the performance of the DNN model. To simulate various crash positions that would happen on the road, we offset the initial position of the cars transversely with different center line distances and rotate the cars with different crash angles, as shown as in Fig. 4. The values of the offset distances are set to 0 m, 0.25 m, 0.5 m, 0.75 m, 1.0 m, 1.25 m and 1.5 m. The values of angles are set to $0^\circ, 15^\circ, 30^\circ$ and 45°. Also, each car was initialized with seven different crash speeds, which are 10 km/h, 20 km/h, 30 km/h, 40 km/h, 50 km/h, 60 km/h and 70 km/h. The combinations of the car speeds are listed in Table 3. To prevent the DNN model to make decision based on numerical artifacts such as FEM mesh differences between the cars instead of the physically-based plastic deformation, the FEM models of two cars were made exactly the same to each other.

Since the FEM models for the two collision cars are exactly the same, thus the collision process of for the speed of the vehicle one is 20 km/h, and that of the vehicle two is 70 km/h is the same as the speed of the vehicle two is 70 km/h and that of the vehicle one is 20 km/h. Thus, the two cases were treated as one case. Therefore, by combining the different offset distances, angles and speeds one by one, there were totally 932 crashing cases that were generated, which are the red and blue colored cases in Table 3. Note that we discard the collision case 10 km/h −0 km/h, which, we think, does not produce enough plastic deformation. The simulations began at the final stage before crashing when the distance between two cars was 2 mm, and ended in 0.2 s after crash when the cars were completely separated. For each collision case, we have 79, 112 FEM mesh points (each car has 39556 FEM nodal points), and each nodal point has residual displacements along three different directions and one effective plastic strain (the von Mises strain). Thus, the total number of displacements and effective plastic strain at those 79112 mesh points are the input data to the DNN model that we established, which are obtained from each collision experiment. Therefore, for the training data, the total number of input data of the neural network is: $952 \times 79112 \times 4$.

3.3. Feature selection

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction (Bermingham et al., 2015). Feature selection techniques are used for four reasons: simplification of models to make them easier to interpret by researchers/users; shorter training times; to avoid the curse of dimensionality and enhanced generalization by reducing overfitting (James et al., 2013; Bermingham et al., 2015).

The central premise when using a feature selection technique is that the data contains some features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information (Bermingham et al., 2015). Redundant and irrelevant are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated (Guyon and Elisseeff, 2003). In this study, plastic strain, residual deformation and residual displacement are all chosen to be features separately and machine learning simulations are performed. Based on the error, residual displacement is the best feature. It is because that it has the largest variance among the three features of data.

3.4. Data filtering

The implemented machine learning algorithm has three deep neural networks. When data flow in-between the different deep neural networks, data filtering is implemented to remove the unnecessary data (Bratko et al., 2006). For instance, after the collision happens in our simulation, there exist lots of points that does not move since they are far from the collision area and remain staying the same position, in other words, they do not have displacement. Therefore, in the first step of the unnecessary data would be removed so our data size can be reduced. Secondly, in a large size database, normally there will be incomplete data. The incomplete data may be outlier values or error data. Hence, data filtering will also be needed for the preprocessing of data. During this procedure, data filtering is the process of getting rid of noise such as outlier values and error data from raw data to make the data clean and be proper for further processing (Wettayaprasit et al., 2007). For instance, if we have a point with relatively large displacement while it is located in an area that should have much smaller displacement, data filter will identify whether it is an outlier. In this study, we filter data based on the result of the previous neural network. There are four benefits of that: 1. Culling. A good filtering tool enables to cull the data and to set it down to a more manageable volume, so that useful data reports can be produced and shared with others. 2. Save and Rerun. The

software products allows to save the filters we create so we can rerun them at a later time when new information is added to the case. 3. Isolate. Pre-search filtering can isolate a subset of records based on fields such as status, evaluation and linked issues, creating slices of data for subsequent processing. 4. Small Production. An effective data set filter allows to narrow down to a smaller set of records for light document production.

3.5. Python implementation

In the developed ANN or DNN algorithm, we use TensorFlow to perform analysis as neural network library. Currently TensorFlow is only supported in Python. In our Python code, we have three hidden layers and the neurons in each layer are $256, 64, 8$ respectively. The learning rate is 0.0035 and the number of steps is $50,000$. The dropout parameter is 0.05 to prevent overfitting. The principle component analysis parameter is 5 and the ratio is 99.9896. A segment of the Python code is shown in Fig. 10.

In constructing the DNN model, the number of hidden layers and the number of neurons in each layer may be varying. We first randomly choose them, and then conduct a few set of computations, using the modified ℓ^2 norm to measure the quality of the predicted pre-crash data until finding the optimal numbers, and finally set the predicted models. The modified ℓ^2 norm is a new metric that we used to design the predictive neural network.

To assess the accuracy of the ANN algorithm, in addition to cross-validation, we calculated the ℓ^2 error norm (Wu et al., 1996; Rizzoli et al., 2012a,b) for the difference between the original and the predicted displacements i.e.

$$e = \|\mathbf{u}^{predict} - \mathbf{u}^{original}\|_{\ell_2} = \sqrt{\sum_{i=1}^{N} \left(\mathbf{u}_i^{predict} - \mathbf{u}_i^{original}\right)^2} \tag{7}$$

where $\left(\mathbf{u}_i^{(1)} - \mathbf{u}_i^{(2)}\right)^2 = \left(u_{xi}^{(1)} - u_{xi}^{(2)}\right)^2 + \left(u_{yi}^{(1)} - u_{yi}^{(2)}\right)^2 + \left(u_{zi}^{(1)} - u_{zi}^{(2)}\right)^2, i = 1, 2, \cdots, N$ are calculated at every nodes of the finite element mesh. The magnitude of ℓ_2 norm is dependent on dimensional unit. To remove of the influence of dimension units, we adopt the following modified ℓ^2 norm:

$$e_{modified} = \sqrt{\frac{1}{A}\sum_{i=1}^{N} \left(\mathbf{u}_i^{predict} - \mathbf{u}_i^{original}\right)^2} \tag{8}$$

where A is the surface area of car model.

4. Material properties

In this study, we selected the material models in LS-DYNA for car structure that are carefully validated and extensively used in car design and crashworthiness analysis in automobile industry.

In specific, the SPCEN cold rolled steel plate, which is commonly used in auto body parts, was chosen as the body material of the car model (Kang et al., 1998). The thixocast A356 alloy, which is also commonly used on car wheels, was chosen as the wheel material (Singh et al., 2014). The strain–stress curves of the two materials are shown in Fig. 11.

The strain and stress curve of the SPCEN and A356 alloy follows the Johnson–Cook constitutive model. The equation of the Johnson–Cook model is as follows:

Table 4
Parameters in the Johnson–Cook model used in the modeling.

Model Part	Car body	Wheels
A(MPa)	208	114.64
B(MPa)	350	35.56
n	0.48	0.0243
C	0.1 4	0.7469
m	0.31	0.0707
Tm	1093.15	973.15
Tr	298.15	298.15
E(Gpa)	210	72.4
Poisson's ratio	0.3	0.33
$\dot{\epsilon}_0$	$1.0\ s^{-1}$	$1.0 \times 10^{-4}\ s^{-1}$
d_1	0.04	0.025
d_2	1.59	16.93
d_3	−6.905	−14.8
d_4	−0.023	0.0214
d_5	1.302	0

$$\sigma = [A + B(\epsilon^p)^n]\left[1 + C\ln\left(\frac{\dot{\epsilon}^p}{\dot{\epsilon}_0}\right)\right]\left[1 - \left(\frac{T - T_r}{T_m - T_r}\right)^n\right] \tag{9}$$

where A, B, C, m, n are material constants; ϵ^p is the plastic strain; $\dot{\epsilon}^p$ is the strain rate; $\dot{\epsilon}_0$ is the reference strain rate; T is the absolute temperature; T_m is the melting temperature and T_r is the reference temperature. The values of the Johnson–Cook model parameters adopted in this model are listed in Table 4.

We adopt the following Johnson–Cook damage model and failure criterion,

$$D = \sum \frac{\Delta\epsilon^p_{eff}}{\epsilon_f} \tag{10}$$

where D is the damage, and

$$\epsilon_f = \left[d_1 + d_2\exp(d_3\frac{p}{\sigma_{eff}}\right]\left(1 + d_4\ln\frac{\dot{\epsilon}_{eff}}{\dot{\epsilon}_0}\right), \tag{11}$$

where d_i are material parameters; $p = -\sigma_{ii}/3$ is the pressure, σ_{eff} is the von-Mises stress or the effective stress, and $\dot{\epsilon}_{eff}$ is the effective (von Mises) strain rate.

5. Results and discussions

We now analyze and discuss the results of machine learning method developed above on its ability to recover pre-crash data and how accurate it is.

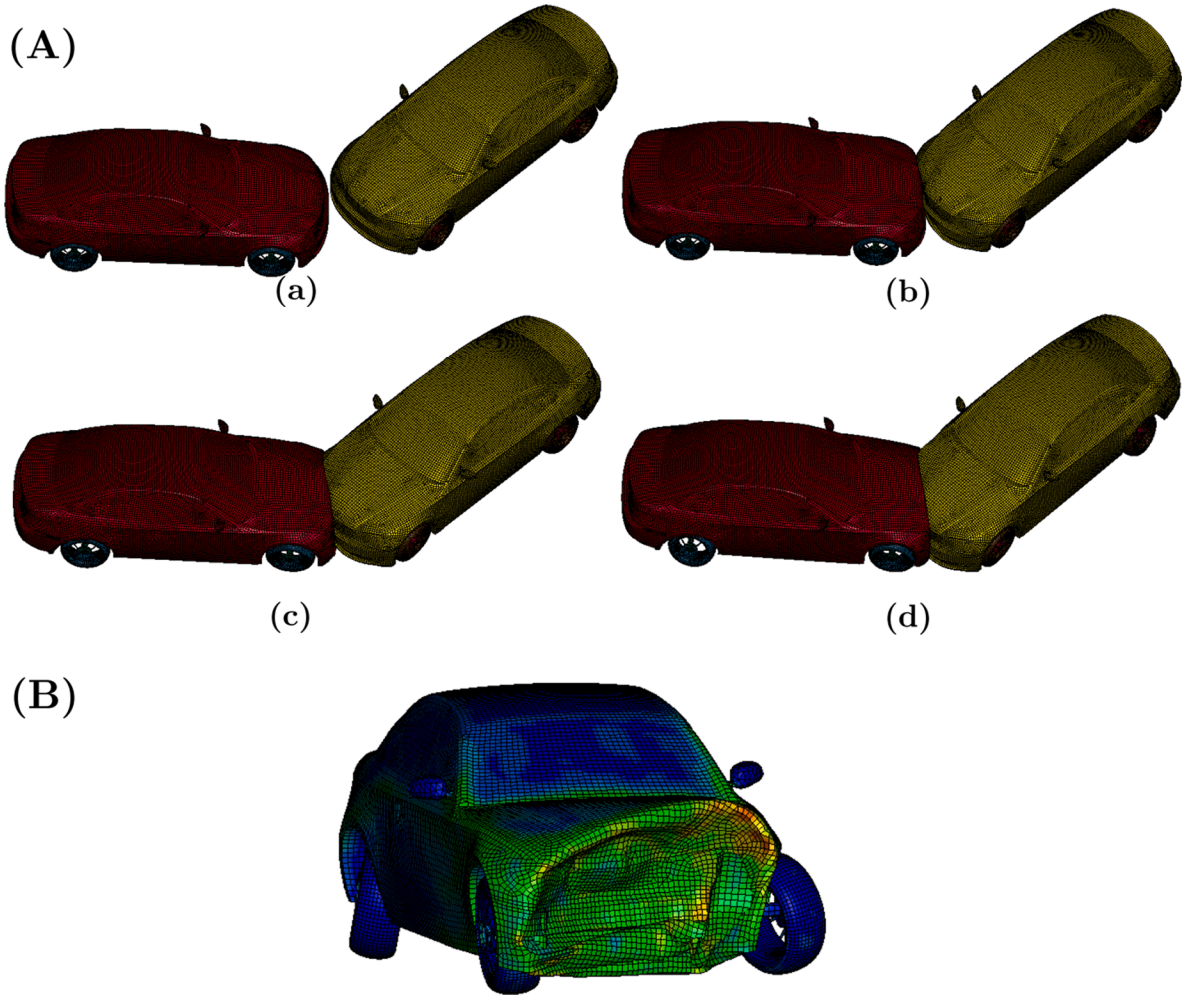

Fig. 5. (A) Sequence of car collisions with an oblique angle (time: 0.04 s - 0.2 s, and (B) Residual plastic strain contour for the left car.

Fig. 7(A) shows one case of animation of car collisions as time goes on with zero angle while Fig. 5(B) shows one case of animation of car collisions as time goes on with non-zero angles. The time ranges from 0.04 s to 0.2 s. During the first 0.04 s period, the two cars are running at their own speeds without any contact or collision.

In Fig. 7, velocities of left and right cars are both 70 km/h and the offset is 0.5 m. Fig. 7(B) shows the plastic strain distribution of the left car after car collision in the zero-angle case. From Figs. 5 and 6, one may find that there is serious damage in the front of the vehicle while the rear of the car only experiences minor or no damage. From the deformation extracted from LS-DYNA, the impact point tends to have larger deformation than other points, a deformation that matches the physical phenomenon happen in real crash event. The center of the damage region is more to the reader's right direction because of the offset influence.

In Fig. 5(A), velocities of left and right car are both 70 km/h, the offset is 1.5 m and rotation angle is 45°. Fig. 5(B) shows the strain of the left car after car collision in the non-zero angle case. From the figure, there is serious damage to the front of the vehicle while the rear of the car only experiences minor or no damage. The hitting point tends to have larger deformation than other points, which match the phenomenon happen in real life. In addition, from the deformation or displacement information, the direction of the impact loads is not parallel to the axis of the car, which is because of the oblique impact instead of direct impact. Moreover, there also exits the rotation of tires.

5.1. Cross-validation

The first ANN model that we built has only one hidden layer with 256 neurons. However, the total loss during the training process could not decrease too much, which indicated the underfitting. Based on try-and-error, a new ANN model with three hidden layers and the sizes of $256, 64, 8$ is built. We used 10-fold cross-validation to calculate the error of the prediction. The average error of offset is 1.58%, the average error of velocities is 2.30%, and the average error of angle is 0.47%. The three errors are relatively low indicating that there is no overfitting in the process.

After training the selected neural network model with all the data, we obtain the finalized model. 50 different sets (around 5% of size of training data) of test data which are not included in the training data are fed to the finalized model to validate its prediction performance and accuracy. Part of the results are shown in Table 5 (result of offset), Table 6 (result of velocities) and Table 7 (result of angles).

The left animation in Fig. 8 shows the simulation of car collisions as time goes on with zero angle. The time ranges from 0.04 s to 0.2 s. The velocities of left and right car are 75 km/h and 25 km/h; and the offset is 0.5 m. In validation, we first build up a testing car model labeled as the original model, and then for a given pre-crash parameters we use LS-DYNA to perform the crash simulations to collect post-crash data, i.e. permanent deformation. After the residual displacement is extracted from the original car model, we feed the residual plastic displacements extracted from LS-DYNA output as the input to the trained machine learning neural network model to predict the pre-crash data, e.g. offset, velocities and angles. The predicted values are 0.50 m offset, 74.14 km/h and 25.04 km/h velocities and 0 angles. Based on the predicted pre-crash data or parameters, we can conduct a new crash test by using LS-DYNA again. We label the new or reconstructed test model as the predicted model. In this reconstructed model, we use the same materials parameters and mechanical methods that are used in the original model. The right animation in Fig. 8 shows the simulation of car

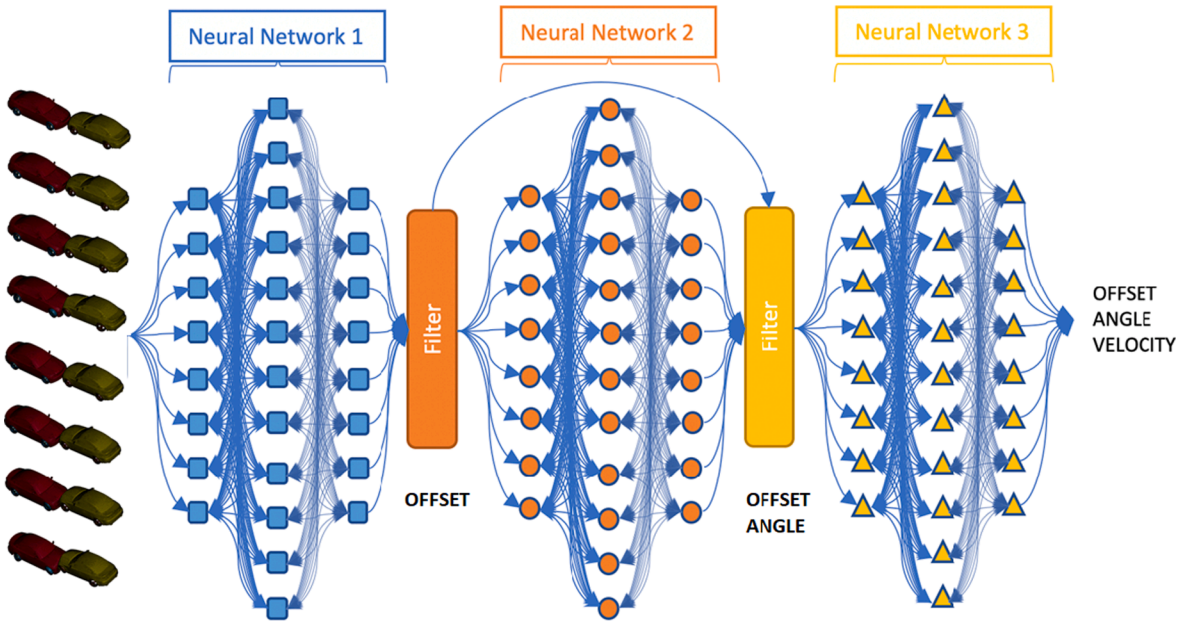

Fig. 6. Flowchart of pre-crash data recovery algorithm.

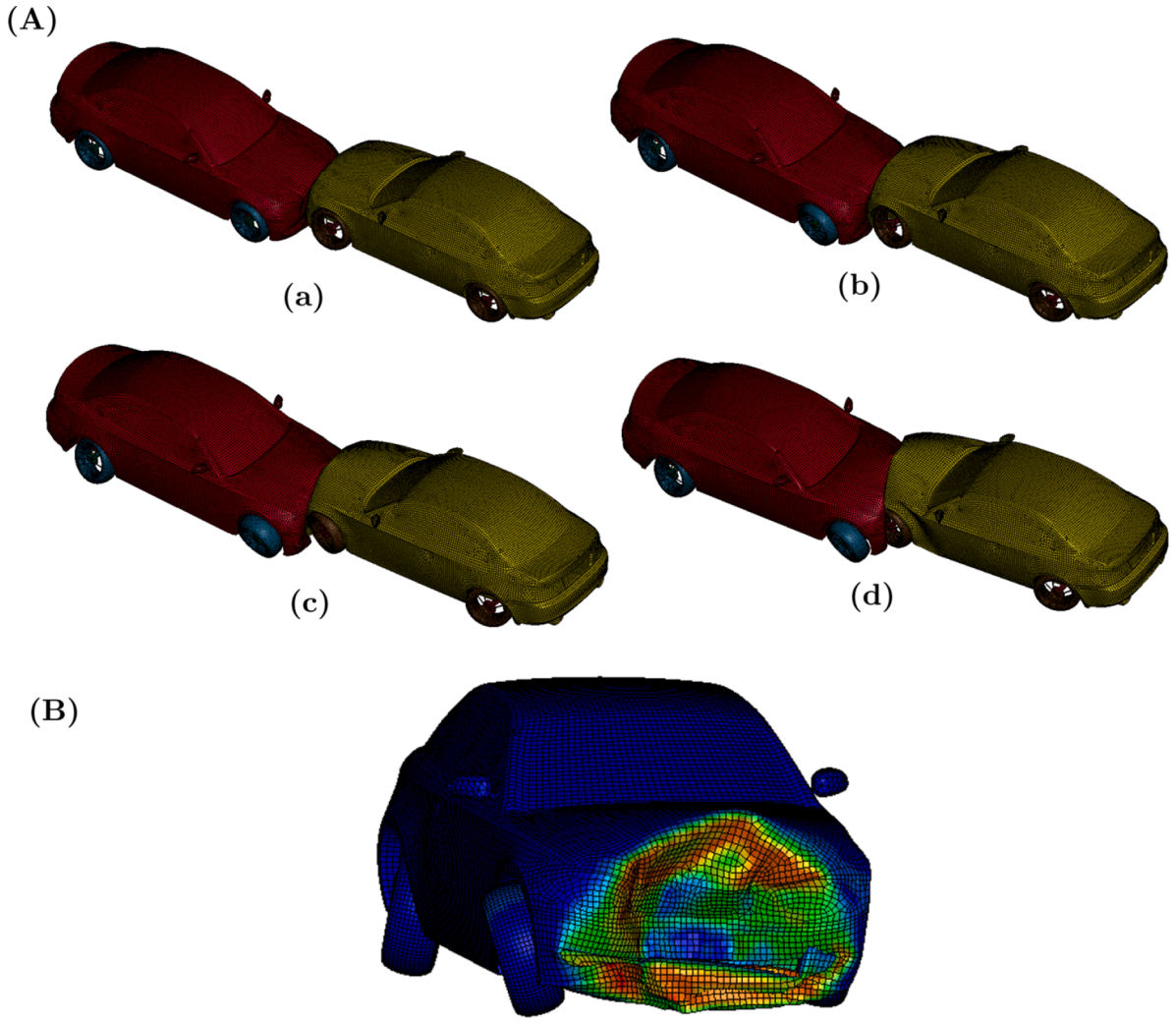

Fig. 7. (A) sequence of head-on (zero initial angle) car collision (time: 0.04 s - 0.2 s), and (B) Residual plastic strain of the left car after car collision.

Table 5
Results of the initial gap between two cars (offset).

Testing cases	1	2	3	4	5
True Value (cm)	50	100	50	100	100
Predict Value (cm)	51.411	100.615	51.411	99.507	97.642
Error(%)	2.823	0.615	2.823	0.492	2.357

Table 6
Results of impact velocities.

Testing cases	1	2	3	4	5
Offset	0.5 m	0.5 m	1.0 m	1.0 m	1.0 m
Car number	Car1-Car2	Car1-Car2	Car1-Car2	Car1-Car2	Car1-Car2
True Value (km/h)	35 - 15	75 - 25	35 - 15	55 - 45	75 - 15
Predict Value (km/h)	35.15-14.97	74.14-25.04	34.38-14.93	55.15-44.43	74.70-14.92
Error (%)	0.45 - 0.18	1.14 - 0.17	1.76 - 0.48	0.28 - 1.26	0.39 - 0.48

Table 7
Results of different impact angles.

Testing cases	1	2	3	4	5	6
True Value (degree)	15	15	30	30	45	45
Predict Value (degree)	15.09	15.09	29.91	29.87	45.12	44.89
Error(%)	0.579	0.579	0.330	0.423	0.258	0.234

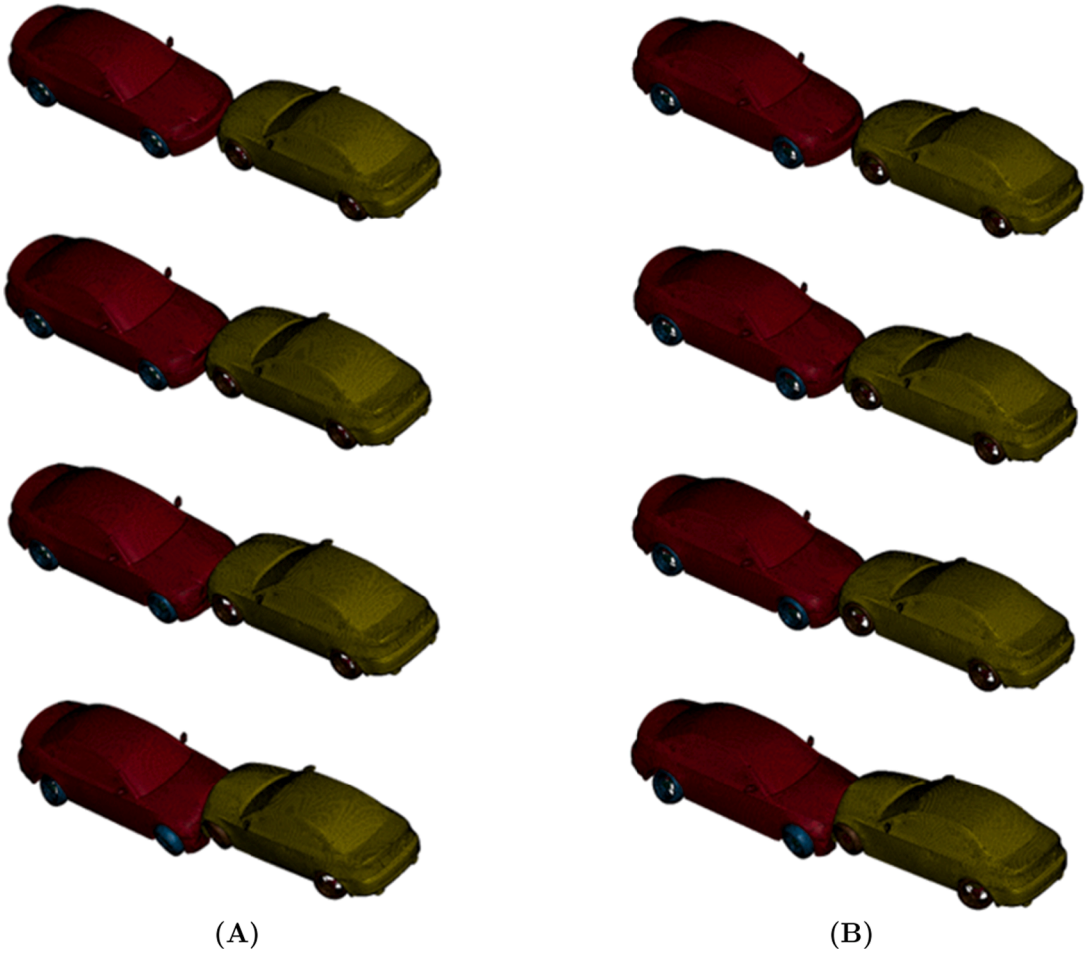

Fig. 8. Comparison between the predicted and original car crash process: (A) Original car crash, and (B) Predict crash.

collisions of the predicted model as time goes on. The two crash tests have also been made in animation movies (see Video 1 in Supplementary information for the original model and Video 2 in Supplementary information for the predicted model).

Fig. 12(a) and (b) show the comparison of the residual displacements of the left and right car between the original test model and predicted test model after car collisions. The left two plots are from the original test model while the right two plots are from the predicted test model. From Figs. 8 and 12(a) and (b), we can find that the prediction of our machine learning model is very promising. To numerically measure the difference, we use modified ℓ^2 norm. The detailed calculation method is explained in the next sections. The modified ℓ^2 norm is 1.78% for the left car and 3.64% for the right car. From these two numbers, we can tell the predicated model is very similar to the original test model, which indicates a good prediction ability of the machine learning neural network.

5.2. Uniqueness of inverse problem

The inverse problem refers to using the results of actual observations to infer the values of the parameters characterizing the initial or boundary conditions. Inverse problems is difficult to solve because first different values of the model parameters may not be consistent with the data, and second finding the values of the model parameters may require the exploration of a huge parameter space. Moreover, if the outcome of the physical system is not sensitive to its initial condition, a same set of the initial conditions may

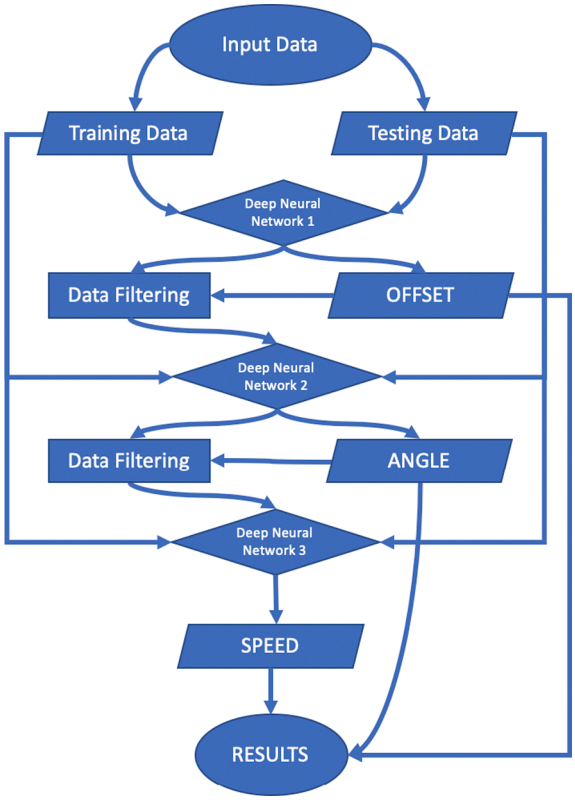

Fig. 9. The flowchart of DNN algorithm.

correspond to multiple similar outcomes in the final observation data set. Thus, for a general inverse problem, one does not always have a unique solution. That is a final solution of an engineering problem may not always uniquely correspond to a unique set of the initial-boundary conditions. Moreover in many applications, the optimization problem that is associated with the inverse problem could be ill-posed.

However, for physical modeling based numerical simulations, a unique set of initial-boundary conditions may have a unique solution for a macroscale problem, if the mathematical model, e.g. a partial differential equation, is not ill-cconditioned or chaotic. The main discovery of the present work is that the final deformation state of a crashed car is so rich and contains so much information, which has almost infinitely many degrees of freedom, if we only take an essential part of it, e.g. residual plastic deformation of the car at a set of discrete points, we can in fact uniquely determine the initial impact parameters in a practically manner. For example, practically, we can use machine learning approach to find a unique solution within an error tolerance of engineering computations. In the practical car crash forensic analysis, one can scan the damaged car components and use a recently proposed AI method developed by the present authors (Wang et al., 2020), and obtain the permanent deformation data at any material points as many as one desires.

To demonstrate the uniqueness of the machine learning inverse solution, we present an illustration example problem, in which we consider two scenarios of a car collision with the same relative impact velocity 45 km/h, to compare differences in two solutions. By fixing the impact offset $(1\ \mathrm{m})$ and impact angle (30°) parameters, we consider two different cases that have the same relative impact velocity i.e. 45 km/h but have different individual car speeds, which are not in the training data set:

*Case*1: Velocity of the left car $=$ 22.5 km/h and Velocity of the right car $=$ 22.5 km/h, and.
*Case*2: Velocity of the left car $=$ 23 km/h and Velocity of the right car $=$ 22 km/h.

We then use the initial conditions to perform the virtual car crash simulations, and subsequently use the simulated post crash data, i.e. plastic deformation, as the input to the developed DNN code to predict pre-crash data. In both cases, we recovered pre-impact velocity based on the DNN prediction:

*PredictedCase*1: Velocity of the left car $=$ 22.5 km/h and Velocity of the right car $=$ 23.67 km/h, and.
*PredictedCase*2: Velocity of the left car $=$ 24.37 km/h and Velocity of the right car $=$ 23.37 km/h,

The calculated ℓ_2 norms for residual displacement are as follows,

```
dataset = tf.data.Dataset.from_tensor_slices((np.array(train_x), np.array(train_y)))
dataset2 = dataset.shuffle(1000).repeat().batch(batch_size)
global_step = tf.Variable(0, trainable=False)

X = tf.placeholder("float", [None, num_input])
Y = tf.placeholder("float", [None, label_dimension])

weights = {
    'w1': tf.Variable(tf.truncated_normal([num_input, n_hidden_1], seed = seed)),
    'w2': tf.Variable(tf.truncated_normal([n_hidden_1, n_hidden_2], seed = seed)),
    #'w3': tf.Variable(tf.random_normal([n_hidden_2, n_hidden_3], seed = seed)),
    'out': tf.Variable(tf.truncated_normal([n_hidden_2, label_dimension], seed = seed))
}

biases = {
    'b1': tf.Variable(tf.truncated_normal([n_hidden_1], seed = seed)),
    'b2': tf.Variable(tf.truncated_normal([n_hidden_2], seed = seed)),
    #'b3': tf.Variable(tf.truncated_normal([n_hidden_3], seed = seed)),
    'out': tf.Variable(tf.truncated_normal([label_dimension], seed = seed))
}

def neural_net(x):
    layer_1 = tf.add(tf.matmul(x, weights['w1']), biases['b1'])
    layer_2 = tf.add(tf.matmul(activation_fn(layer_1), weights['w2']), biases['b2'])
    #layer_3 = tf.add(tf.matmul(activation_fn(layer_2), weights['w3']), biases['b3'])
    out_layer = tf.matmul(activation_fn(layer_2),weights['out']) + biases['out']
    return out_layer

logits = neural_net(X)
loss_op = tf.reduce_mean(tf.abs(logits-Y)/Y, axis = 0)
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
train_op = optimizer.minimize(loss_op)


init = tf.global_variables_initializer()

config = tf.ConfigProto(device_count={"CPU": 8})
sess = tf.Session()

sess.run(init)

for step in range(1, steps+1):
    batches = sess.run(dataset2.make_one_shot_iterator().get_next())
    batch_x, batch_y = batches[0], batches[1]
    sess.run(train_op, feed_dict={X: batch_x, Y: batch_y})
    if step == 1 or step % display_step == 0:
        loss, err = sess.run([loss_op, err_rate], feed_dict={X: batch_x,
                                                             Y: batch_y})
        print("Step " + str(step) + \
            ", Loss= " + str(loss.round(3)))#+ \


results = sess.run(logits, feed_dict = {X: predict_x, Y: predict_y})
pred_errs = sess.run(separate_err, feed_dict={X: predict_x, Y: predict_y})
```

Fig. 10. Python code segment of DNN pre-crash data recovery algorithm.

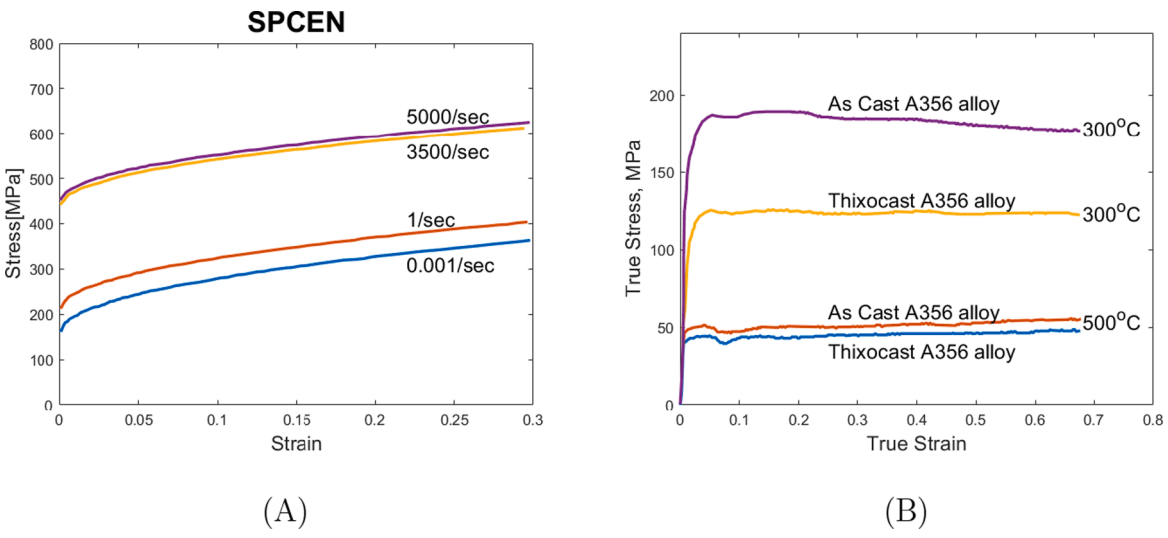

Fig. 11. (A) Strain–stress curve of the SPCEN steel, and (B) Strain–stress curve of the thixocast A356 alloy.

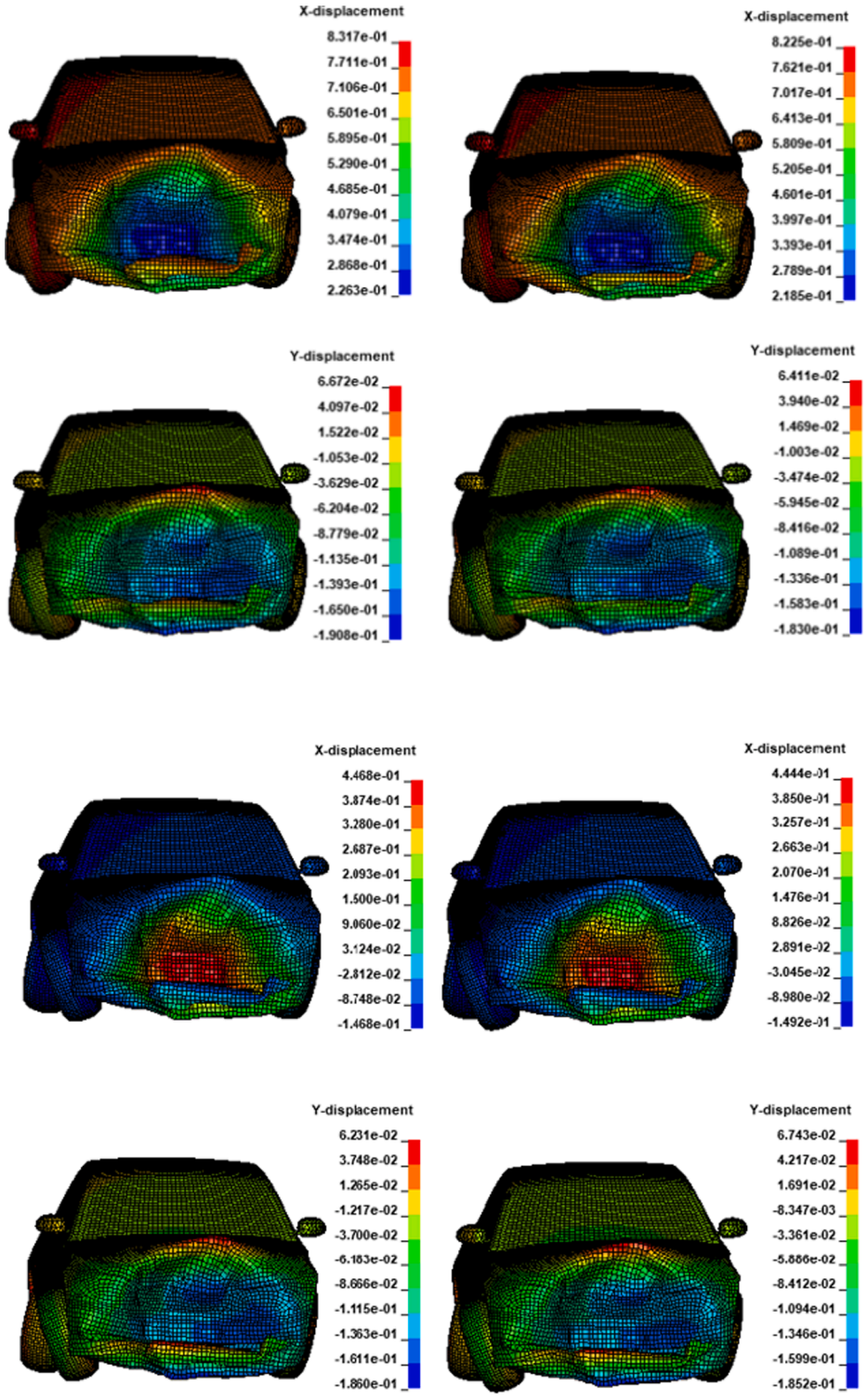

Fig. 12. Comparison of the residual displacement of the crashed cars between the original test model (left) and predicted test model by using parameters from LY-DYNA (right) after car collision.

ℓ_2-error of $\|Case1 - PredictedCase1\|$ is 2.06%;
ℓ_2-error of $\|Case2 - PredictedCase1\|$ is 2.09%;
ℓ_2-error of $\|Case1 - PredictedCase2\|$ is 0.26%, and.
ℓ_2-error of $\|Case2 - PredictedCase2\|$ is 0.21%.

The l_2 errors are computed by using Eq. (8). In car crash forensic analysis, if the relative impact velocities of two accidents are the same, and the individual car speeds are very close, it is usually difficult to tell the differences between two car accidents. In other words, one may say that the solution is not unique. Suppose that we may not know which car corresponds to which predicted solution, or in other words, one may mistaken the crash accident 1 as the crash accident 2, i.e. the solution may not be unique. Then by examining the l_2 errors of the four different scenarios, from the above results we can see that the ℓ_2-norm error of $\|Case1 - PredictedCase1\|$ is always smaller than that of $\|Case1 - PredictedCase2\|$ ($2.06\% < 2.09\%$) while the ℓ_2-norm error of $\|Case2 - PredictedCase2\|$ is always smaller than that of $\|Case2 - PredictedCase1\|$, i.e. $0.21\% < 0.26\%$. These indicate the uniqueness of the machine learning approach, at least numerically.

During car crash, the permanent plastic deformation usually depends on the relative impact velocity of the two impacting cars. From this example, one can clearly see that under the same relative impact velocity, the developed machine learning algorithm can correctly predict the pre-crash velocity for each individual car, which means that the AI algorithm can find the unique solution from tiny differences in permanent plastic deformation of the collision cars.

5.3. Activation functions

To predict pre-crash data, in this work, we used three hidden layers and the same training data, and apply different activation functions to train the model to predict offset. For the pre-crash offset prediction, the results are shown in Table 8. Among these five activation functions, ReLU is the best activation function in terms of error. The average error of offset is 1.28%, while the other methods all get around 4% error. The same phenomenon happens during the process to predict velocities and angles.

Moreover, the ReLU activation function takes relatively less time than other activation functions. Finishing a 10-fold cross validation simulation, ReLU model uses around 2824.50 seconds in a 10-core computer, while it takes around 1932.06 seconds for sigmoid method, 2848.56 seconds for tanh method, 3182.80 seconds for softmax method and 2854.70 seconds for ReLU Square method. This is because the other methods take more time to calculate the gradient during the back propagation process, especially it becomes a very complex equation to calculate the derivative of softmax method, while the gradient of ReLU is very easy to calculate, because that is just a Heaviside step function.

From this derivative equation, we can see clearly that ReLU has a simpler gradient. The biggest advantage of ReLU is its non-saturation gradient, which greatly accelerates the convergence of stochastic gradient descent method compared with the sigmoid and tanh functions. However, this is not its only advantage, and another desired property is that it has fast and simple calculation. In fact, the ReLU gradient can be implemented by simply thresholding a matrix of activations at zero. This is in sharp contrast to those expensive calculations/operations involved with tanh/sigmoid neurons, e.g. exponentials, etc.

In addition, interesting enough, we find that the method with the smallest total loss does not always lead to the least error, for example when using the sigmoid activation function. A possible cause of small training error while large test error is overfitting. On the other hand, the largest total loss also does not lead to the least error either, such as the cases that softmax and ReLU Square methods were used. It may be caused by underfitting, in which both the training and test errors are large. Overall, we find that the training error by using ReLU activation function is relatively small, which also leads to the smallest test error.

5.4. Data filtering

Without data filtering, the machine learning algorithm is still able to predict pre-crash parameters. An ANN machine learning model with three hidden layers and sizes of $256, 64, 8$ is developed to predict offset, velocity and angles at the same time. The same training data set is used. After 10-fold cross-validation, the average errors of the prediction are 3.49%, 15.25% and 11.30% separately for the three parameters.

However, if we apply data filtering, the average error of offset reduces to 1.58%, the average error of velocities reduce to 2.30%, and the average error of angle reduces to 0.47%. The accuracy of offset, velocity and angles get improved by around 2%, 13% and 8%.

From the comparison, we find that by dividing one neural network to several small ones and predicting output step by step we can decrease the error significantly. This is because that between each step data filtering is necessary to avoid un-necessary data and noise.

Table 8
Comparison of different activation functions.

Activation Function	Error (%)	Training Time (seconds)	Total Loss
ReLU	1.28	2824.50	48.68
Sigmoid	4.029	2932.06	21.81
Tanh	4.062	2848.56	21.56
Softmax	4.77	3182.80	203.52
ReLU Square	4.398	2854.70	219.47

5.5. Cross-validation

In this study, we use cross-validation to evaluate the machine learning model. Cross-validation, sometimes called rotation estimation, or out-of-sample testing, is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set (Kohavi, 1995). It is mainly used in settings where the goal is prediction, and one may want to estimate how accurately a predictive model will perform in practice. In using data-driven machine-learning algorithm to prediction unknown features or parameters, that total data set is usually divided into two parts: *the training data set* and *the test data set*. The data in the training data set is used to training and subsequently build the neural network, and the data in the test data set or the validation data set is used to test or validate the neural network that is built. The goal of cross-validation is to test the model's ability to predict parameters based on new data that was not used in building the model, which is an important process that can identify or flag problems like overfitting or selection bias and provide an insight on how the model can be extended to independent datasets i.e., an unknown data set, or the data set from a real physical problem (Tou and Gonzalez, 1974; Cawley and Talbot, 2010).

Cross-validation combines and averages measures of fitness in prediction leading to a more accurate estimate of model prediction performance. It helps us evaluate the quality of the computation model. Moreover it helps us select the model that will perform best on unseen data and avoid overfitting and underfitting. One of the popular cross validation examples is k-fold validation method.

In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subset samples as shown in Fig. 13. Of the k subset samples, a single subset sample is retained as the validation data for testing the model, and the remaining $k-1$ subset samples are used as training data. The cross-validation process is then repeated k times, with each of the k subset samples used exactly once as the validation data. The k results can then be averaged to produce a single estimation (Zhang, 1993; Fushiki, 2011).

5.6. Validation tests for predictability

To validate the predictability of the deep neural network machine learning model developed in this work, or the prediction accuracy on the data that are out of the parameter range of training data, we also conduct a validation test study on the predictability of the developed DNN algorithm.

In doing so, we created a $32 \times 2 \times 5 = 320$ data set as the training data, in which there are 32 sets of impact velocity combinations (see blue color pairs in Table 3), 5 offset parameters $0m$, 0.25 m, 0.5 m, 0.75 m and 1 m, and 2 impact angle parameters 0° and 15°. We selected eight different collision cases whose data are partially or completely beyond the parameter range of the above training data set. To check whether or not our model is sufficiently good to predict the pre-crash collision conditions, we employed the 320 neuron DNN model to perform predictions on the selected eight test data. However, for all eight extrapolation cases, the offsets are still within the width length of the vehicle, which is 1.783 m (see Table 1), the impact angles are not beyond 90°, and the initial velocity for each vehicle does not pass 140 km/h. Table 9 shows the exact initial collision conditions of the different cases.

Table 10 is the predicted results by using the developed DNN model and their relative errors, one can see that, all the errors for pre-crash velocity predictions are relatively low, except for Case 7 and Case 8; and all the errors for pre-crash angle are high. This is not a surprise, since the all the test angles, and impact velocities in Case 7 and Case 8 are significantly beyond the parameter range of the training data.

It is noted that the pair of Case 4 and Case 6 and the pair of Case 5 and Case 7 have the same offset, angle, and the same relative impact velocity. The only difference is that for Cases 4–6, they have different absolute car speed, 35–40 (km/h) and 15–60 (km/h); and for Cases 5–7, they have different absolute car speed, 40–40 (km/h) and 20–60 (km/h). From Table 10, one can see that the machine learning algorithm still provide unique solutions for extrapolated prediction results. That is the machine learning neural network inverse solution is practically unique, and it can recognize the difference in the impact speed of individual car. It may be noted that the

Fig. 13. Overview of k-fold cross-validation method (Claesen et al., 2014).

Table 9
Parameters of validation test cases.

Parameters	Offset(m)	Angle(°)	Velocity(km/h)
Case 1	0.81	19	27–39
Case 2	1.16	25	34–12
Case 3	1.34	33	13–45
Case 4	1.60	50	35–40
Case 5	1.70	58	40–40
Case 6	1.60	50	15–60
Case 7	1.70	58	20–60
Case 8	0	15	40–110

Table 10
Prediction results for 8 extrapolation testing data sets.

Cases	Offset(m)	Error(%)	Angle(°)	Error(%)	Velocity(km/h)	Error(%)
Case 1	0.87	7.41	19.9	4.74	27.88–36.92	3.26–5.33
Case 2	1.09	6.03	25.6	2.4	34.74–12.87	2.18–7.25
Case 3	1.29	3.51	31	6.06	13.52–42.11	4.00–6.42
Case 4	1.49	6.88	44.7	10.6	34.28–39.54	2.05–1.15
Case 5	1.50	11.76	45	22.41	39.22–41.04	1.95–2.60
Case 6	1.49	6.88	44.7	10.6	15.77–55.12	5.13–8.13
Case 7	1.50	11.76	45	22.41	21.33–57.34	6.65–4.43
Case 8	0.04	0.04	15.09	0.6	37.68–100.66	5.80–8.49

above calculations were conducted in a small 320 neuron DNN model instead of the original 952 neuron DNN code. If we use the original 952 neuron DNN code, we can significantly reduce the error, because many cases would become interpolation cases. Nevertheless, in overall, the predicted results are still acceptable in practice.

Based on the above results of the test cases, we think that the developed DNN model has some ability to conduct extrapolated prediction on the pre-crash impact condition.

6. Conclusions

In this work, we have developed a machine learning based framework, i.e. an artificial intelligence DNN algorithm and related pre-crash data recovery methods to identify and recover the precise initial impact conditions for different of crash scenarios and different car structures in an inverse manner, based on the residual plastic deformation as forensic signatures.

Even though mathematically we cannot prove that the artificial intelligence-based inverse solution is theoretically unique, based on our computational results, we think that the AI-based inverse problem solution is "practically" unique in the range of the data set acquired and in the sense of engineering modeling and experimentation. In other words, we find that the information contained in the residual plastic deformation distribution of each crashed car is so rich, any change of pre-crash data or impact conditions will produce different, indelible, and unique signature of residual plastic deformation, which can be readily detected and captured by the machine learning based approach or AI methods. On the other hand, our simulation results suggested that the final damage state of the car structure is not sensitively dependent on pre-crash condition, such that it may renders chaotic solutions.

An inverse problem may have multiple solutions, i.e., it is not mathematically unique. The scenarios of the inverse problems that have multiple solutions are two: (1) micro-mechanics problems, in which a macroscale state corresponds to multiple microscale states, which is not the case for pre-crash impact condition prediction studied here, and (2) the initial impact condition is not sensitive to the measured data of damage state, so that multiple damage states may correspond to a same set of initial conditions.

However, what we have found in this research is that for car crash problem it is involved with much plastic energy dissipation in many parts of the vehicle during the car structure damage process, which renders the problem solution sensitively depending on the initial impact conditions but nevertheless not chaotic, meaning that slight perturbations on impact conditions will generate indeleble and stable differences in crash outcomes. Thus, we can use the final crashed states of the car to trace and identify the initial impact and collision condition if the crash process is highly energy dissipative, meaning that plastic deformation data points are overwhelmingly rich, which can capture any small differences in pre-crash impact conditions. It is shown in numerical modeling and simulation that if the material of the car components is elasto-viscoelastic, the machine learning algorithm to accurately predict the initial collision state based on the final structure damage state. The mathematical ill-conditioning or the non-uniqueness of the initial impact conditions reconstruction problem is lost or contained by the passive plastic deformation. In other words, even though there may be multiple initial collision condition solutions for final crashed car state, the differences of these multiple sets of initial collision conditions are unnoticeable in the accuracy of engineering measurement process. Therefore, we may say that the inverse problem has practically unique solution, if the material damage process is highly dissipative, which justifies the machine learning approach by using the final car structure damage state to accurately find the initial collision conditions.

The main advantage or contribution of the proposed machine learning approach is that one can accurately determine and recover

pre-crash data by simply measuring residual permanent deformation of crashed cars in the crash site. It is true that so far we only used the virtual data (FEM data) to train the deep learning neural network, this is because firstly the work present here is a proof-of-concept; secondly the cost of the virtual data in low, and thirdly, today almost of all passage vehicles' design, manufacture, and crashworthiness analysis are mainly based on modeling and simulation. The next step of this research is to developed a hybrid ANN model that uses the a small set of experimental data to adjust virtual data based ANN model, and use the real crash data to validate the hybrid ANN model, before it can be commercialized.

Another novelty of this work is that we divide the main neural network into three smaller ones to predict the pre-crash data in a convoluted sequential manner, and we also employed ℓ^2 norm as the metric to measure the error of the machine learning model, and incorporate it into ANN algorithm to exam the uniqueness of the algorithm. This development may have significant impacts on car accident forensic analysis and structure failure analysis and structure reliability analysis (Nowak and Collins, 2012), and it provides a powerful tool for material and structure forensic diagnosis, determination, and identification of damage loading conditions in accidental failure events, such as car crashes and infrastructure or building structure collapses.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

SL would like to acknowledge the support from Livermore Software Technology (ANSYS-LST), which provided five licences LS-DYNA software packages conducting all finite element simulations reported in this work. The authors also would like to thank Berkeley Research Computing (BRC) center for providing high-performance computer cluster, Savio, to perform most of computation tasks reported in this work.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.trc.2021.103009.

References

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000. Artificial neural networks in hydrology. I: Preliminary concepts. J. Hydrologic Eng. 5(2), 115–123.
Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., 2007. Greedy layer-wise training of deep networks. Adv. Neural Informat. Process. Syst. 153–160.
Bermingham, M.L., Pong-Wong, R., Spiliopoulou, A., Hayward, C., Rudan, I., Campbell, H., Wright, A.F., Wilson, J.F., Agakov, F., Navarro, P., Haley, C.S., 2015. Application of high-dimensional feature selection: evaluation for genomic prediction in man. Sci. Rep. 5, 10312.
Bohn, B., Garcke, J., Iza-Teran, R., Paprotny, A., Peherstorfer, B., Schepsmeier, U., Thole, C.A., 2013. Analysis of car crash simulation data with nonlinear machine learning methods. Procedia Comput. Sci. 18, 621–630.
Bratko, A., Cormack, G.V., Filipič, B., Lynam, T.R., Zupan, B., 2006. Spam filtering using statistical data compression models. J. Machine Learn. Res. 7 (Dec), 2673–2698.
Cawley, G.C., Talbot, N.L., 2010. On over-fitting in model selection and subsequent selection bias in performance evaluation. J. Machine Learn. Res. 11, 2079.
Chen, G., Li, T., Chen, Q., Ren, S., Wang, C., Li, S., 2019. Application of deep learning neural network to identify collision load conditions based on permanent plastic deformation of shell structures. Comput. Mech. 64, 435–449.
Chong, M.M., Abraham, A., Paprzycki, M., 2005. Traffic accident analysis using machine learning paradigms. Informatica 29 (1).
Claesen, M., Simm, J., Popovic, D., Moor, B., 2014. Hyperparameter tuning in Python using Optunity. In: Proceedings of the International Workshop on Technical Computing for Machine Learning and Mathematical Engineering, vol. 1, pp. 3.
De Villiers, J., Barnard, E., 1993. Back propagation neural nets with one and two hidden layers. IEEE Trans. Neural Networks 4, 136.
Fushiki, T., 2011. Estimation of prediction error by using K-fold cross-validation. Stat. Comput. 21, 137.
Greenwood, J., Williamson, J.P., 1966. Contact of nominally flat surfaces. Proc. Roy. Soc. London. Series A. Mathe. Phys. Sci. 295, 300.
Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. J. Machine Learn. Res. 3, 1157.
Haining, R., 1993. Spatial Data Analysis in the Social and Environmental Sciences. Cambridge University Press.
Hallquist, J.O., 2007. LS-DYNA Keyword Users Manual. Livermore Software Technology Corporation, p. 970.
Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning. Springer.
Haug, E., Scharnhorst, T., Du Bois, P., 1986. FEM-Crash, Berechnung eines Fahrzeugfrontalaufpralls. VDI Berichte 613, 479–505.
Jain, A.K., Mao, J., Mohiuddin, K.K., 1996. Artificial neural networks: A tutorial. Computer 29 (3), 31–44.
James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An Introduction to Statistical Learning, vol. 112. Springer.
Johnson, G.R., Cook, W.H., 1985. Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures. Eng. Fracture 21, 31–48.
Jones, A., Keatley, A.C., Goulermas, J.Y., Scott, T.B., Turner, P., Awbery, R., Stapleton, M., 2018. Machine learning techniques to repurpose Uranium Ore Concentrate (UOC) industrial records and their application to nuclear forensic investigation. Appl. Geochem. 91, 221–227.
Kang, W., Cho, S., Huh, H., Chung, D., 1998. Identification of dynamic behavior of sheet metals for an auto-body with tension split hopkinson bar. Tech. Rep., SAE Technical Paper.
Kirchdoerfer, T., Ortiz, M., 2018. Data-driven computing in dynamics. Int. J. Numer. Meth. Eng. 113, 1697.
Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. Ijcai 14 (2), 1137–1145.
König, A. et al., 2011. Knowledge-Based and Intelligent Information and Engineering Systems, Part II: 15th International Conference, KES 2011, Kaiserslautern, Germany, September 12-14, 2011, Proceedings, vol. 6882, Springer.

Lei, X., Liu, C., Du, Z., Zhang, W., Guo, X., 2019. Machine learning-driven real-time topology optimization under moving morphable component-based framework. J. Appl. Mech. 86, 011004.
Mena, J., 2016. Machine Learning Forensics for Law Enforcement, Security, and Intelligence. Auerbach Publications.
Nasrabadi, N.M., 2007. Pattern recognition and machine learning. J. Electron. Imag. 16 (4), 049901.
Nowak, A.S., Collins, K.R., 2012. Reliability of Structures. CRC Press.
O'Boyle, N.M., Banck, M., James, C.A., Morley, C., Vandermeersch, T., Hutchison, G.R., 2011. Open Babel: An open chemical toolbox. J. Cheminformat. 3, 33.
Ren, S., Chen, G., Li, T., Chen, Q., Li, S., 2018. A deep learning-based computational algorithm for identifying damage load condition: An artificial intelligence inverse problem solution for failure analysis. Comput. Model. Sci. Eng. (CMES) 117 (3), 287–307.
Rizzoli, P., Bräutigam, B., Kraus, T., Martone, M., Krieger, G., 2012a. Relative height error analysis of TanDEM-X elevation data. ISPRS J. Photogramm. Remote Sens. 73, 30.
Rizzoli, P., Bräutigam, B., Kraus, T., Martone, M., Krieger, G., 2012b. Relative height error analysis of TanDEM-X elevation data. ISPRS J. Photogramm. Remote Sens. 73, 30.
Sajda, P., 2006. Machine learning for detection and diagnosis of disease. Annu. Rev. Biomed. Eng. 8, 537–565.
Sebastiani, F., 2002. Machine learning in automated text categorization. ACM Comput. Surv. (CSUR) 34 (1), 1–47.
Singh, S.K., Chattopadhyay, K., Phanikumar, G., Dutta, P., 2014. Experimental and numerical studies on friction welding of thixocast A356 aluminum alloy. Acta Mater. 73, 177.
Sohn, S.Y., Lee, S.H., 2003. Data fusion, ensemble and clustering to improve the classification accuracy for the severity of road traffic accidents, in Korea. Saf. Sci. 4 (1), 1–14.
Specht, D.F., 1990. Probabilistic neural networks. Neural Networks 3, 109.
Specht, D.F., 1991. A general regression neural network. IEEE Trans. Neural Networks 2, 568.
Steps To Follow After A Car Collision, Motorward: Articles/Guides July 9 2018. https://www.motorward.com/2018/07/steps-to-follow-after-a-car-collision/.
Tou, J.T., Gonzalez, R.C., 1974. Pattern recognition principles.
Wang, C., Li, S., Zeng, D., Zhu, X., 2020. Quantification and compensation of thermal distortion in additive manufacturing: A computational statistics approach. Comput. Methods Appl. Mech. Eng. 375, 113611.
Wettayaprasit, W., Laosen, N., Chevakidagarn, S., 2007. Data filtering technique for neural networks forecasting. In: Proceedings of the 7th WSEAS International Conference on Simulation, Modelling and Optimization (World Scientific and Engineering Academy and Society (WSEAS), pp. 225–230.
Wu, F., Yang, X.H., Packard, A., Becker, G., 1996. Induced L2-norm control for LPV systems with bounded parameter variation rates. Int. J. Robust Nonlinear Control 6, 983.
Zhang, P., 1993. Model selection via multifold cross validation. The Annals of Statistics, pp. 299–313.
Zheng, X., Zheng, P., Zhang, R.-Z., 2018. Machine learning material properties from the periodic table using convolutional neural networks. Chem. Sci. 9, 8426.
Zhou, Q., Tang, P., Liu, S., Pan, J., Yan, Q., Zhang, S.C., 2018. Learning atoms for materials discovery. Proc. Nat. Acad. Sci. 115, E6411.