Level-set and Learn: Convolutional neural network for classification of elements to identify an arbitrary number of voids in a 2D solid using elastic waves

Fazle Mahdi Pranto[1], Shashwat Maharjan[2], and Chanseok Jeong[3,4]

[1]Research assistant, School of Engineering and Technology, Central Michigan University, Mount Pleasant, MI 48859, USA. Email: prant1f@cmich.edu

[2]Research assistant, School of Engineering and Technology, Central Michigan University, Mount Pleasant, MI 48859, USA. Email: mahar1s@cmich.edu

[3]Assistant Professor, School of Engineering and Technology, Central Michigan University, Mount Pleasant, MI 48859, USA.

[4]Member, Institute for Great Lakes Research, Central Michigan University, Mount Pleasant, MI 48859, USA (corresponding author) Email: jeong1c@cmich.edu

Keywords — Void detection, Level-set method, Element-wise classification, Ultrasonic non-destructive test (NDT), Machine learning, Convolutional neural network, Inverse-scattering problem.

ABSTRACT

We present a new convolutional neural network (CNN)-based element-wise classification method to detect a random number of voids with arbitrary shapes in a 2D plain strain solid subjected to elastodynamics. We consider that an elastic wave source excites the solid including a random number of voids, and wave responses are measured by sensors placed around the solid. We present a CNN for resolving the inverse problem, which is formulated as an element-wise classification problem. The CNN is trained to classify each element into a regular or void element from measured wave signals. Element-wise binary classification enables the identification of targeted voids of any

shapes and any number without prior knowledge or hint about their locations, shape types, and numbers while existing methods rely on such prior information.

To this end, we generate training data consisting of input-layer features (i.e., measured wave signals at sensors) and output-layer features (i.e., element types of all elements). When the training data are generated, we utilize the level-set method to avoid an expensive re-meshing process, which is otherwise needed for each different configuration of voids. We also analyze how effectively the CNN performs on blind test data from a non-level-set wave solver that explicitly models the boundary of voids using an unstructured, fine mesh. Numerical results show that the suggested approach can detect the locations, shapes, and sizes of multiple elliptical and circular voids in the 2D solid domain in the test data set as well as a blind test data set.

INTRODUCTION

Defects of various sizes and shapes—such as voids, inclusions, and cracks—can compromise the structural integrity of civil and mechanical structures. The literature has shown several types of nondestructive testing (NDT) approaches, such as ultrasonic testing, for identifying structural defects (Hellier 2013). Characterizing such defects using wave-based empirical NDT procedures without any systematic numerical method is time-consuming, requires a trained technician, and is only applicable to simple problems (e.g., a problem where only the location of a single line crack of an assumed orientation is detected). Thus, systematic inverse modeling is required to detect complex defects subjected to dynamic excitation, and, in general, a series of computational dynamic forward problems with varying locations and sizes of the scatterers are iteratively solved during the inversion process. In such inverse modeling procedures, the boundary element approach (BEM) has been a prominent forward modeling tool because the discretization of only the boundaries of voids and cracks (Wrobel 2002; Jeong et al. 2009) allows the BEM's computing cost to be multiple order of magnitude lower than the finite element method (FEM), and re-discretization of the boundary is straightforward. Because the BEM approach employs the Green's function in an elastodynamic medium, as its major disadvantage, BEM can hardly be applicable in arbitrarily-heterogeneous media of which Green's function is not straightforward to compute (Guzina and Pak 1996). On the other hand, the extended finite element method (XFEM) and the level-set method have been adopted for the inverse-scattering problem to overcome this limitation of BEM because (*i*) they model heterogeneous medium with scatterers straightforwardly (Sukumar et al. 2004; Ashari and Mohammadi 2011; Elguedj et al. 2009; Menouillard et al. 2010), and (*ii*) unlike the standard FEM, neither the XFEM nor the level-set method necessitates extensive re-meshing for modeling varying shapes of scatterers in the iterative process of solving forward problems.

Optimization methods, hinged on a number of forward modeling iterations, have been used with the XFEM and the level-set method for resolving inverse-scattering problems as briefly shown in the following. Rabinovich et al. (2007) and Rabinovich et al. (2009) identified cracks using the XFEM and the genetic algorithm (GA) in 2D structures under both static and dynamic excitations. Waisman et al. (2010) studied the performance of the GA inversion modeling under elastostatic conditions for detecting various types of structural damage, such as cracks and holes with regular and irregular shapes. Chatzi et al. (2011) presented a novel GA, coupled with a generic XFEM and the level-set modeling of an elliptical void to model cracks or voids of any shape, that avoids entrapment in local optima. They also showed the experimental validation of the numerical method for detecting an arbitrary crack in a 2D plate. Jung et al. (2013), Jung and Taciroglu (2014), and Jung and Taciroglu (2016) examined a new method to identify cracks and voids in a heterogeneous medium using elastodynamic waves by combining the dynamic XFEM and the level-set method with a gradient-based search algorithm. In addition, they proposed the cubic spline method for discretizing the boundary of arbitrarily-shaped scatterers and a divide-and-conquer strategy to tackle the solution multiplicity. Sun et al. (2013) and Sun et al. (2014) detected multiple flaws using the XFEM as well as the level-set method and employed new topological variables in the optimization process to activate and deactivate defects during the analysis along with an enhanced artificial bee colony technique to tackle the non-uniqueness of the considered inverse problem. Nanthakumar et al. (2013) suggested a new multilevel coordinate search strategy to detect elliptical voids in piezoelectric structures. Yan et al. (2015) suggested a guided Bayesian inference method, combined with the XFEM, to identify and determine multiple cracks in elastic structures without prior knowledge of the number of cracks. Zhang et al. (2016) suggested a method that uses the dynamic XFEM for modeling a fracture in elastodynamic medium and uses Nelder-Mead and Quasi-Newton optimization methods to identify cracks in plates. Wang and Waisman (2017) suggested a new crack-tip enrichment function for the XFEM to detect cracks in bimaterials. Livani et al. (2018)

used a new approach using both the Extended Spectral Finite Element method and the particle swarm optimization (PSO) algorithm to detect multiple cracks. Khatir and Wahab (2019) combined PSO and Jaya optimizers with the XFEM and extended isogeometric analysis for detecting cracks in plane structures. Zhang et al. (2019) proposed a method for identifying voids in a continuous medium utilizing time-domain dynamic response from the level set method and the analytically-calculated shape derivative of an objective function. Ma et al. (2020) proposed a new method using the XFEM and an improved artificial bee colony algorithm that can identify multiple cracks without any prior knowledge of the flaws. Fathi et al. (2021) proposed the dynamic XFEM with an enhanced vibrating particles system (EVPS) to solve an inverse-scattering problem. In all of the aforementioned methods, the time-consuming iteration-based optimization methods have been used for the inverse modeling of scatterers. Since optimization involves a series of forward modeling after measurement data are fed into the process, it is not possible to detect scatterers quickly (e.g., in a second) from measurement data by using such optimization-based methods.

On the other hand, a few papers have recently shown that machine learning (ML) would overcome such limitation (i.e., long computing time) of the optimization-based methods for the inverse-scattering problem. ML has been studied for the inverse-scattering problems because of its potentially-short computing time (e.g., less than a second), once training is done, and its potentially-high accuracy even when the number of control parameters is large (e.g., thousands or more). However, to date, the related literature is still relatively thin, and the latest, published studies in the ML-based inverse-scattering are still in early stage as shown in the following. Jiang et al. (2021) combined the level-set method and the extreme learning machine (ELM) to detect voids of a known number in 2D solid structures. They modeled circular or elliptical voids during the process to generate the training data for an elastonynamic wave-based NDT. However, for their method, the number of voids and their shape types (circular or elliptical) should be *a-priori* known to the simulator. Thus, the method was limited only for identifying the coordinates of the centroids and radii (or major/minor axes with orientations) of circular (or elliptical) voids provided that their numbers and shape types are known in advance. Jiang et al. (2022) also extended the approach for identifying structural flaws in thin structures by integrating the scaled boundary finite element method (SBFEM) into the data generation process for a CNN. They considered Lamb waves in thin plates, and damage types (e.g., surface cracks of simple wedge shapes on a boundary of the thin plate) are *a-priori* known in the simulation. Then, from measured wave data, their trained CNN determined only the number of cracks, only from 1 to 3, and, in turn, determined three shape parameters of each crack (e.g., the depth, the wedge angle, and the width of a crack). Gao et al. (2022) trained a fully connected neural network, from synthetic electromagnetic wave training data, to identify a scattering object in an application like a radar problem. They considered the 2D/3D Helmholtz equation and electromagnetic wave system, and their trained neural networks identify the shape parameters of a single targeted scattering object of a complex shape (e.g., a star shape). However, their work is limited by the assumption that there should be only one scattering object, of which approximate location should be *a-priori* known, in a domain. To overcome the limitations (i.e., requiring such *a-priori* known information) of the aforementioned ML methods for the inverse-scattering problems, one should study a new ML-based approach for identifying the locations and shapes of an arbitrary number of voids in a solid without *a-priori* known information.

To the best of our knowledge, there has been no study, using an artificial neural network (ANN), for identifying the locations and shapes of voids of an arbitrary number in a solid by using elastodynamic waves. To fill this research gap, this paper presents a new data-informed CNN method, hinged on element-wise classification, to identify an arbitrary number of voids in a 2D plain strain solid, of which elastodynamic wave responses are measured by sensors placed around the solid. We use the level set method to avoid time-consuming re-meshing for various configurations of voids, which are iteratively updated while training data are generated. We generate training data that consist of input-layer features (i.e., measured signals) and output-layer features (i.e., the element types of all elements). The CNN is trained to classify the type of an element (i.e., void or non-void) from measured wave signals at sensors. Thus, the trained CNN results in a contour map of the element-wise classification, which shows the map of the probability for each element to be a void element. Thus, from the contour map, an engineer could infer the locations, sizes, and shapes of an arbitrary number of voids in the 2D plain-strain domain. This research serves as the prototype in a 2D setting, and it can be extended for detecting voids of arbitrary shapes and numbers in a 3D solid.

FORWARD MODELING

Governing Equation

This paper considers a domain $\Omega \subset R^2$ (see Fig. 1), which is occupied by a homogeneous linear elastic undamped solid medium with voids. The governing equation of the displacement field of elastic waves in the solid can be expressed as:

$$\nabla \cdot \boldsymbol{\sigma} = \rho \ddot{\mathbf{u}} \quad \text{in } \Omega, \tag{1}$$

where the displacement field of a vector wave is $\mathbf{u} := \mathbf{u}(x, y, t) = [u_x, u_y]^T$; $\boldsymbol{\sigma} := \boldsymbol{\sigma}(x, y, t)$ denotes Cauchy stress tensor; $\nabla \cdot (\ )$ denotes the divergence operator; $\rho := \rho(x, y)$ denotes the mass density. The wave responses are subject to the at-rest initial conditions. The displacement field $\mathbf{u}$ vanishes on Γ_u, and the boundary of a void Γ_h is characterized by the traction-free condition while the prescribed traction is applied on Γ_n. We note that, while elastodynamic waves for imaging the solid are generated and propagated, the material property of the solid or the boundary of voids are not altered. Namely, the presented elastodynamic wave-based imaging is performed after targeted voids are created. The elastodynamic wave-based imaging while a void or crack is formed is beyond the scope of this paper. For such a problem, we refer to (but not limited to) Stanchits et al. (2015) who demonstrated a passive wave-based method (e.g., acoustic emission) used for characterizing the initiation and propagation of fractures in geological rocks induced by hydraulic fracturing.

Level-set approximation and the FEM

This section revisits the conventional level-set method (Chatzi et al. 2011; Jung et al. 2013; Sun et al. 2014) to model a void on a background mesh and compute wave responses in a solid domain with voids. This work uses the FEM wave solver, which is based on the level-set approximation, because the level-set solver does not necessitate the onerous re-meshing whenever the geometry of voids is updated in each training data set.

In the weak form of the governing equation, the displacement field $\mathbf{u}$ is approximated using the finite element approximation and an enrichment function $V(x, y)$. Namely, $\mathbf{u}$ in an element is approximated as:

$$\mathbf{u}^h(x, y, t) = V(x, y) \sum_{i=1}^{N} \phi_i(x, y) \mathbf{u}_i(t), \tag{2}$$

where

$$V(x,y) = \begin{cases} 1, & \text{if} \quad x, y \in \text{an element classified as a non-void element,} \\ 0, & \text{if} \quad x, y \in \text{an element classified as a void element.} \end{cases} \tag{3}$$

The approximation in Eq. (2) consists of nodal displacement $\mathbf{u}_i$ and local shape function $\phi_i(x,y)$ at the i-th node, and we use 9-node quadrilateral elements in this paper.

If the centroid and 4 other nodes (among total 9 nodes) of an element are inside the actual boundary of a void, that element is considered as a void element. Otherwise, an element is defined as a non-void element. The exemplary illustration of such categorization of elements can be seen in Fig. 2.

The weak form and Eq. (2) lead to the time-dependent discrete form in each element. It should be noted that, if an element is classified as a void element, its element stiffness and mass matrices become zero. After the global assembly of matrices and a force vector, we obtain the time dependent equation in terms of a global displacement solution vector. We adopted the Newmark time integration method (Newmark 1959) to solve for the global solution vector for each discrete time step.

Verification of the level-set wave solver

Prior to our investigation on the performance of the presented CNN-based inverse-scattering modeling, we verify our in-house level-set forward solver, written in MATLAB, by comparing our wave responses with the reference solution obtained by using the Finite Element Analysis Program (FEAP) (Taylor 2017), which uses an explicit mesh for the same domain (see Fig. 3). The mesh is generated using MESH2D, a Delauny mesh generator (Engwirda 2005; Engwirda 2014). In our level-set-based forward wave solver, the domain is discretized by using a standard background mesh of 9-node square elements with an element size of 0.01 m, while an unstructured mesh of 3-node triangular elements with targeted edge lengths of 0.004 m is used in FEAP.

This verification considers a homogeneous square-shaped solid domain, of which extent is 0.8 $\times$ 0.8 m and includes a single circular void of 0.1 m radius, located at the center of the domain. For both models used by the level-set solver and FEAP, we use material properties of aluminum with Young's modulus (E) of 71.5 $\times 10^9$ Pa, mass density (ρ) of 2800 kg/m^3, and Poisson's ratio (ν) of

0.33. The compressional wave velocity (v_p) and the shear wave velocity (v_s) are calculated as:

$$v_p = \sqrt{\frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}}, \quad v_s = \sqrt{\frac{E}{2\rho(1+\nu)}}, \tag{4}$$

so that the values of v_p and v_s are 6151 ms^{-1} and 3098 ms^{-1}, respectively. The domain is subject to a traction-free condition on the left, top, and right surfaces, and it is constrained by a fixed boundary condition on the bottom surface.

A point wave source is located at $x = 0.3$ m and $y = 0.8$ m, at which the following Ricker wave signal (Fig. 5), with a peak amplitude of 5,000 N/m and a dominant frequency of 20,000 Hz, is applied:

$$P(t) = \begin{cases} \frac{-5000((0.25.e^{-0.25(2\pi ft-3\sqrt{6})^2}.(2\pi ft-3\sqrt{6})^2)-0.5)-13.e^{-13.5}}{0.5+13.e^{-13.5}}, & \text{if} \quad t \leq \frac{6\sqrt{6}}{2\pi f}, \\ 0, & \text{if} \quad t > \frac{6\sqrt{6}}{2\pi f}, \end{cases} \tag{5}$$

where f denotes the central frequency of the Ricker signal.

Fig. 4 shows the snapshots of the amplitudes of displacement at 7μs, 10 μs, 12 μs, 14 μs, 16 μs and 20 μs obtained by our forward wave solver. Fig. 6 illustrates a comparison of u_y, at the receivers' locations, computed by our level-set wave solver and FEAP. Here, sensors are placed at the left, top, and right surfaces with a sensor spacing of 0.1 m, except the corners and the source location (0.3, 0.8) m, and they are numbered clockwise from the bottom left at (0, 0.1) m to the bottom right at (0.8, 0.1) m. Both signal data, shown in Fig. 6, respectively from our level-set solver and FEAP, agree with each other very well, implying that the level-set solver described is verified and can be utilized for the data generation.

Data Generation and Randomizer

Data generation for machine learning simulation necessitates solving a forward problem considering voids whose number and geometries change per each data set. This work uses a homogeneous 2D square-shaped domain of 0.8 m $\times$ 0.8 m in a plane-strain setting discretized by an element size of 0.02 m. Namely, the entire domain is discretized into a structured background mesh with a total of 1600 square elements.

A schematic representation of the boundary conditions for the data generation is shown in Fig. 7. The domain is subjected to traction-free conditions on the left, top, and right surfaces and the fixed boundary condition on the bottom. The wave sources are loated at $(0.3, 0.8)$ m, $(0, 0.3)$ m, and $(0.8, 0.4)$ m. The Ricker signal $P(t)$ in Eq. (5) is applied as a nodal force signal in a manner such that its positive value points toward the inside of the domain at all source locations. The central frequency of the Ricker time signal at all the sources is f = 20,000 Hz, and its maximum amplitude is 5,000 N/m.

The solid is made of the same aluminum as the one used for the aforementioned verification with elastic modulus $E = 71.5$ GPa, mass density $\rho =$ 2,800 kg/m^3, and Poisson's ratio $\nu = 0.33$. The total simulation time for each forward iteration is 1,000 μs, and the time-step is 1 μs. Here, a total of 18 sensors are evenly placed on all the sides, except the bottom, at a spacing of 10 mm.

We use our randomizer to generate a random number of elliptical-shaped voids of random sizes and locations within the domain. Among all possible common types of void shapes (e.g., triangles, trapezoids, potatoes, boomerangs, etc.), we choose to use a thin elliptical shape because of the following reasons. First, it is more straightforward to parameterize an elliptical shape than others. Second, by utilizing only thin ellipses, we can populate void elements in a manner such that void elements, within one or multiple thin ellipses, mimic wide cracks or interconnected wide cracks. Similarly, previous works (Chatzi et al. 2011; Sun et al. 2014) have also utilized voids of only elliptical shapes for testing their proposed structural damage-detection algorithms. The equation for defining the actual boundary of each elliptical void is:

$$\frac{[(x - x_0)\cos(\alpha) + (y - y_0)\sin(\alpha)]^2}{m^2} + \frac{[(x - x_0)\sin(\alpha) - (y - y_0)\cos(\alpha)]^2}{n^2} = 1, \tag{6}$$

where $(x_0,\ y_0)$ denotes the center of an ellipse; m and n are the lengths of its major and minor axes, respectively; and α is the angle of the major axis with respect to the x-axis (see Fig. 8a).

In our data set, we generate ellipses with a major axis ranging from 8 mm to 15 mm, a minor axis ranging from 4 mm to 8 mm, and an orientation ranging from $5°$ to $175°$. In order to generate unbiased training data, we randomly locate our voids all over the domain. A sample of our randomly generated elliptical voids in the domain is shown in Fig. 8b. When each data set is generated, the following steps are carried out:

- The positions, sizes, and number of the voids are randomly updated.
- Per the updated void geometry, we save the data indicating which element is categorized as void or not per our level-set approximation. If an element is classified as a void element, we assign a value of one at the element, and if it is categorized as a non-void element, we assign a value of zero. The assigned-value data for all the 1,600 elements will be used for element-by-element classification in our trained CNN.
- We run our level-set wave solver and save the displacement field of wave responses measured at the sensor locations.
- Out of 36,000 training data, our randomizer generates 6 sets of 6,000 training data sets, each of which accounts for 0, 1, 2, 3, 4, and 5 voids, respectively.

Our pseudo code of the data generation is as follows:

Randomizer, using the 2D level-set-based wave solver, to generate data sets.

```
***FOR*** an iteration index: $1 \rightarrow$ (the total number of data sets= 36,000)
    $\Rightarrow$ Randomly set the values of the parameters of up to 5 elliptical voids in the domain within
       the following ranges:
            $0.05 \leq m \leq 0.15$ m;
            $0.010 \leq n \leq 0.015$ m;
            $m \leq x_0 \leq (0.8 - m)$ m;
            $n \leq y_0 \leq (0.8 - n)$ m;
            $5 \leq \theta \leq 175$ deg.
    $\Rightarrow$ Update mass and stiffness matrices per the level-set approximation by using the above
       elliptical void parameters.
    $\Rightarrow$ Solve the 2D wave propagation problem.
    $\Rightarrow$ Save the displacement data at every three-time step from 18 sensor locations as the input-
       layer feature.
    $\Rightarrow$ Save element-wise classification data (1 for void and 0 for no-void) of all the 1,600 elements
       as the output-layer feature. ***ENDFOR***
```

Fig. 9 shows a heat map, where the number in its color bar indicates the number of data sets in which a given element is recognized as a void element by our level-set approximation during the generation of 36,000 data sets. Fig. 9 presents that our randomizer, in general, covers nearly the entire domain, to promote unbiased learning of our CNN, except the areas around the surrounding boundaries (Γ_u and Γ_n) because our randomizer prevents ellipses from being located on Γ_u and Γ_n. Our randomizer follows a uniform distribution, and all the possible values of the parameters of our elliptical voids are likely to fall under the uniform distribution during the data generation process. Per the law of large numbers, the heat map may look more uniformly than the presented one in Fig. 9, except the areas near the boundaries, as the number of data sets increases further from the presented 36,000 sets.

Our randomizer creates a random set of ellipses in each data set in a manner such that the values used for all the elliptical parameters are likely to be independent from each other. Although the areas of multiple voids may overlap with each other, we do not double count void elements. Because our neural network is aimed at identifying targeted void elements regardless of possible overlapping of ellipses in training data sets, such potential overlapping does not affect our prediction performance.

Data Preparation

A total of 36,000 data sets of input- and output-layer feature data are provided to train the CNN. The input-layer feature data contain displacement values from 18 sensors in two directions, u_x and u_y, each of which includes 334 data (simulation data from 0 to 1,000 ms are saved at every 3 ms). The output-layer feature data consist of a serialized combination of binary values of either 0 or 1, for all the elements, where 0 represents a non-void element, and 1 represents the presence of a void element.

The training samples are then split into three parts: (*i*) training, (*ii*) validation, and (*iii*) test data sets. Out of 36,000 samples, we separate 30,000 data sets for training, 1,000 data sets are for validation, and the remaining 5,000 data sets are for testing. The training data sets are used to expose the CNN to learn important input- and output-layer data feature relations. The validation subset is set up to keep track of the performance of the CNN during training. The test subset is used to evaluate the CNN's performance.

Sample input-layer feature data in Fig. 10a shows that the range of input-layer feature data is in the order of 10^{-8}, which makes it difficult for the CNN to learn critical input- and output-layer data feature. By using mean, maximum, and minimum values as statistical parameters, the input data range is normalized to the values between -1 and 1. The mean, maximum, and the minimum values for the normalization are calculated from the training data set for a fairer analysis on the validation, test, and the blind test data sets than otherwise. The following equation is used to normalize the input displacement data:

$$A^n_{ijk} = \frac{A_{ijk} - A^{\text{mean}}_{\text{train}}}{A^{\text{max}}_{\text{train}} - A^{\text{min}}_{\text{train}}}, \tag{7}$$

where A^n_{ijk} is the normalized value of the displacement data set including the training, validation or test set; the subscripts i, j, and k correspond to the k-th time-step of j-th channel of the i-th sample; A_{ijk} is the un-normalized value of the input data matrix; $A^{\text{mean}}_{\text{train}}$ is input data matrix training data set's mean value; and $A^{\text{max}}_{\text{train}}$ and $A^{\text{min}}_{\text{train}}$ are input data matrix training data set's maximum and minimum values, respectively. The ordinate-axis in Fig. 10b clearly shows the normalization, and this normalized data set is then passed as input to the neural network architecture.

THE ARCHITECTURE OF THE CNN

The problem of interest is developed as a binary classification problem where the input data are processed through the CNN to yield predictions of each element as either void or non-void. During the training process, we use a binary cross-entropy (BCE) as our loss function, comparing the CNN-predicted probability of each element to be a void to its targeted counterpart:

$$\mathcal{L} = -\frac{1}{M}\sum_{i=1}^{M}\left(\frac{1}{N}\sum_{i=1}^{N} E_i \log(P(E_i)) + (1-E_i)\log(1-P(E_i))\right), \tag{8}$$

where E_i is the targeted label ("1" for voids and "0" for non-void elements) of the i-th element; N is the total number of elements; M is the total number of the training data; and $P(E_i)$ is the predicted probability (ranged from zero to one) of the i-th element being void.

Among other types of ANNs (e.g., deep and graphical neural networks), we choose a CNN due to its remarkable feature extraction and learning capabilities. We present a novel CNN that can be applied to time-signal series at multiple sensors, where each signal is 1D data in time. CNN has unique layers called convolutional layers, where the input data are transformed using filters before being passed onto the next layer. The overall algorithm of our CNN architecture is presented in Fig. 11, which consists of a single convolutional layer, a Max Pooling layer, a Flatten layer, a fully-connected hidden layer, and the output layer. The convolutional layer consists of multiple filters of fixed filter sizes which convolve over the provided multi-channel input data to extract the feature patterns. Here, 47 filters convolute through the sensor channels of a single data set resulting in 47 Intermediate Feature Maps (see the bottom part of Fig. 12), which results in an accumulation of 47 convoluted data vectors of 334 time steps for each channel. The convoluted data are then passed through a non-linear activation function to produce an intermediate output on which a Max Pooling layer is applied to extract significant feature maps from the previous convolutional layer. The feature map from the Max Pooling layer is a multi-dimensional matrix and is transformed into a one-dimensional vector using the Flatten layer. The output from the Flatten layer undergoes a series of weights and bias computation in a fully-connected hidden and an output layer. The detail of each step in the CNN is shown in the following so that the readers can replicate and verify the presented CNN.

The design of a convolutional layer

The convolutional operation—the beginning part of the CNN (see the dashed box in Fig. 11)—is presented in Fig. 12. We use a convolutional layer with 47 filters. Each filter is a vector of 19 components, each of which is a single-valued number whose values are initialized using the “Xavier” initialization (Glorot and Bengio 2010). Every CNN filter is unique as each filter value is initialized randomly (Maharjan et al. 2022). Padding is added to the input data to preserve the input length dimension, and the convoluted values are passed through Leaky Rectified Linear Unit (LReLU) activation function. The LReLU activation function is defined as:

$$r_i = f_{\mathrm{LReLU}}(a_i) = \begin{cases} h \cdot a_i & \text{if } a_i < 0, \\ a_i & \text{if } a_i \geq 0, \end{cases} \tag{9}$$

where r_i is the activation function-applied outcome; a_i is the input of the activation function, such as $i-$th component of an array of the convoluted values shown in Fig. 12; and h is a fixed value (0.3 in this work). The initialized filter 1 (out of total 47 filters in this work) slides across the length of the input data set in a sliding-window dot product operation as illustrated in Fig. 12. The remaining 35 channels from the training data set undergo a similar operation for the filter 1. The 35 signals, after the convolution, are added into one signal of the same number of time-steps (334 in this work) to provide a convoluted output vector of length 334, preserving the input length. This operation is repeated for the remaining 46 filters to produce a feature map of size $(334, 47)$ which is then fed into the Max Pooling Layer.

The design of the final fully-connected layers

In the latter part of the CNN, we use a Max Pooling Layer (see the bottom of Fig. 11) of pool size 4 to extract prominent feature values from the previous layer by taking the maximum value from the four adjacent vector values reducing the dimension of the output vector to a length of (84, 47). The values are then passed onto the Flatten later converting the two-dimensional data matrix in the Max Pooling layer to a one-dimensional data array resulting in a vector of length 3,948. The Flatten layer enables the feeding of the feature-extracted input data from the convolutional operation to a fully-connected layer. The weights and bias are learnable parameters and are applied at the fully connected layer and its following layer as:

$$O_i = \sum_{j=1}^{d} w_{ij} b_j + s_i, \tag{10}$$

where w_{ij} is the weight coefficient between the two consecutive neurons at adjacent layers; b_j is the j-th feature (or neuron) from a previous layer; d is the size of the data from a previous layer; s_i is the bias associated with each neuron; and O_i is the i-th neuron's calculation outcome. The fully-connected layer consists of 3,950 neurons and uses the LReLU function in Eq. (9) as the activation function. Here, the LReLU function takes O_i from Eq. (10) as input and passes the array to the output layer with 1,600 neurons to represent the binary information at all the 1,600 elements in the domain employing the use of the "Sigmoid" activation function:

$$r_i = f_{\text{sigmoid}}(a_i) = \frac{1}{1+e^{-a_i}}, \tag{11}$$

where r_i is the activation function-applied outcome; and a_i is the input value to the $i-$th neuron in the output layer. Namely, the sigmoid activation function in our final output layer produces a probabilistic measure, which is continuously ranged from 0 (when a_i is a negative number of a sufficiently sizable magnitude, e.g., $f_{\text{sigmoid}}(-2) = 0.1192$ and $f_{\text{sigmoid}}(-5) = 0.0067$) to 1 (when a_i is vice versa, e.g., $f_{\text{sigmoid}}(5) = 0.9933$). It is visualized in a contour map, to predict if a particular element out of the 1600 elements could be a void or not. We also attempted to employ regularization methods such as batch normalization and dropout to tackle possible over fitting in the training process. However, the use of such regularization worsened the validation recall. Our CNN takes 130 seconds to train for 50 epochs.

Optimization—the learning

Under the Tensorflow framework, we use the "Adam" optimizer with a learning rate of 0.0005 to learn the values of the filters in the convolutional operator as well as weights and bias in Eq. (10). We trained the CNN for 50 epochs (or iterations) and used a batch size of 550. In the first iteration, the Xavier-initialized unknown parameter values are used to predict an output which is compared against the actual output. Through back propagation and automated differentiation, the unknown parameter values are updated until the specified epoch where the network has effectively learned feature relations between the input- and output-layer features.

NUMERICAL RESULTS

In this section, we show how the CNN loss function and an evaluation metric are updated during the training process. We also present the CNN's performance on (*i*) a test data set generated by the same level-set solver and (*ii*) a blind test data set generated independently using FEAP. We show the performance of our CNN using the following evaluation metrics:

$$\text{accuracy} = \frac{tp+tn}{tp+fn+tn+fp} \times 100\ [\%], \tag{12}$$

$$\text{precision} = \frac{tp}{tp + fp} \times 100\ [\%], \tag{13}$$

$$\text{recall} = \frac{tp}{tp + fn} \times 100\ [\%], \tag{14}$$

$$\text{f1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \times 100\ [\%], \tag{15}$$

where tp, tn, fp, and fn are the number of, respectively, true-positive, true-negative, false-positive, and false-negative assessments of all the elements in a given data set. While evaluation metrics use only binary information (tp, tn, fp, and fn), our CNN's predicted value of each element's type is non-binary, ranged from 0 (non-void) to 1 (void). We round such a non-binary value to a binary value when we calculate the evaluation metrics.

Performance on training and validation data

We generated 36,000 data sets and set aside 30,000, 1,000, and 5,000 data sets for training, validation, and test, respectively. The convergence of the loss function for the training and validation data sets over epochs (i.e, the iteration during the training) is shown in Fig. 13a, and the increase in recall (among the aforementioned metrics) over the epochs is shown in Fig. 13b. In Fig. 13, we observe that our CNN attains the converged values of loss and maximum recall from about 25 epochs. Our optimizer effectively identifies the parameters of the CNN such that the prediction from the CNN matches the targeted counterpart in training and validation data sets. Our CNN predicts the probability for each element to be a void element in a 2D-domain to provide a more thorough and robust prediction. Through this approach, an engineer can use our CNN output to make their own further judgement to characterize voids formed within a domain.

We also present our CNN's best and worst predictions on the test data sets in Figs. 14 and 15, where it is observed that our CNN successfully detects void elements at various locations in the test data set. The evaluation metrics on the test data set are also shown in Table 1. It is shown that, in our test data sets, the CNN is effective in terms of detecting void elements that correspond to thin, elliptical shapes.

Performance of the presented CNN on blind test data from an independent FEM wave solver (FEAP) without the level-set approximation

The aforementioned test data set is generated by the same level-set wave solver that is used for generating the training data set. In this section, we show our CNN's performance on blind test data generated from a wave solver that does not use the level-set approximation but models the boundary of voids using a very fine unstructured, explicit mesh. Namely, the wave response from such a wave solver is more truthful to the real physics than the level-set solver, particularly in terms of modeling the traction-free boundary of a void. However, the FEAP-based measurement data still cannot fully replicate real experiment because it is very challenging to experimentally implement the ideal, fixed bottom boundary condition. The related background of the challenge is shown in a recent experimental-validation work (Lloyd et al. 2023) and summarized at the end of this paper.

Figs. 16 to 18 show that our CNN can detect targeted elliptical voids in the blind tests and show their overall shapes because our neural network is trained by data sets considering voids of elliptical shapes. Figs. 16 to 18 also show that our CNN detects two elliptical voids more accurately than three of them. Accordingly, the evaluation metric is higher in the cases of two elliptical voids than that of three elliptical voids as shown in Table 2. Such a result is due to the increase in complexity in the reconstruction of three elliptical voids compared to one and two circular/elliptical voids.

We also examine, via blind test data, whether our CNN can effectively recognize voids of shapes that deviate from elliptical ones. Namely, our neural network identifies the locations of circular voids as shown in Figs. 19 and 20, but their shapes are reconstructed as the combination of multiple ellipses in a manner such that the predicted ellipses delineate the targeted circular voids. Therefore, the values of recalls for predicting the two circular targeted voids are lower than those for the two elliptical targets as shown in Table 2.

Numerical results using another data set made by elliptical and circular voids

Intrigued by the blind tests for targeted circular voids shown in Figs. 19b and 20b, we hypothesized that adding circular voids and thicker ellipses in training data sets may improve the performance of identifying targeted circular voids.

To test this hypothesis, we generated additional 36,000 data sets with a chance of major axis (m) being equal or close to minor axis (n). To this end, we changed the previously-used range of n from ($0.01 \leq n \leq 0.015$ m) to ($0.02 \leq n \leq 0.08$ m) in Algorithm 1. In the new data set, we observe that 5% of our data include circular voids.

Using our re-trained CNN model, we were successful in significantly improving circular-void reconstruction as shown in Fig. 19c and Fig. 20c and in Table 2. We also noticed better performances in the reconstruction of two ellipses as shown in Fig. 17c and Fig. 16c. In addition, the metrics (except the recall) in Table 2 and Fig. 18c show better reconstructed results of three ellipses for the re-trained CNN.

In short, our result shows that our neural network generally identify the targets more accurately if we increase the variation of void shapes in data sets by incorporating circular as well as thin and thick elliptical voids in them.

DISCUSSION: THE SELECTION OF AN EVALUATION METRIC

In this FEAP-generated blind test data sets, we would like remark the following aspects of the aforementioned metrics from Eq. (12) to (15).

- One metric (i.e., "accuracy") significantly overestimates the performance. "Accuracy" provides an extremely high value because a large portion of the elements (greater than 90%) in the entire domain consists of non-void elements. Thus, tn (true negative) is always high in the presented numerical examples, which overestimates the performance evaluation despite below-average performance on certain test examples.
- Furthermore, the evaluation using "precision" relies on computing the fp (false positive). In our 2D domain, we are challenged with predicting tp (true positive) amidst a large number of tn because of the aforementioned 90% non-void element occupancy, and the order of magnitude of fp is same as that of tp. Thus, "precision", somewhat, underestimates the performance of the CNN model.
- The use of "recall" as the evaluation metric is factored to address the mentioned limitation

of precision that is attributed to fp (false positive).

- “f1-score” is the harmonic mean of precision and recall.

We deem “precision”, “recall”, and “f1-score” fit to provide an unbiased-analysis of the prediction model. Among the three metrics, “recall” is chosen in the convergence test shown in Fig. 13, but the other two can be used as well.

CONCLUSION

We discuss a new method for detecting voids in a 2D plane-strain solid subject to elastic waves. The proposed method employs two components: (a) the level-set method for solving successive forward wave problems to generate training data for varying locations and shapes of elliptical voids and (b) the implementation of a CNN using the training data from the forward solutions. Under the level-set method, we can avoid re-meshing of the domain per the randomly changing boundaries of voids while we generate training data.

Once the CNN is trained, it can effectively classify each element as a void or non-void in the test data set. We also employed blind test data sets from an independent wave solver that does not use the level-set method but models the boundary of voids using a very fine unstructured, explicit mesh. The wave response from the independent wave solver is more truthful to the real physics than the level-set solver, particularly in terms of modeling a void's boundary. Our CNN recognizes the features between the input and output data, leading to effective predictions on the blind test data sets.

Many classification problems in the literature have been centered around classifying only a few objects in image data. For instance, we refer to a problem, where an autonomous vehicle classifies whether detected objects (or labels) in visual data are pedestrians, vehicles, animals, pavement markings, trees, light poles, etc. One prime example is the YOLO-v4 model which has proven effective to conduct precise real-time object detection for multiple classes within a video frame (Bochkovskiy et al. 2020). In contrast, our presented method aims at the classification of 1600 labels, where the order of magnitude of the number of the labels is larger than those in the aforementioned common classification problems. Such a large-scale, element-wise classification has not been reported in the literature. Thus, the methodology and the new findings in this paper will lay a foundation for further related problems—e.g., an element-wise classification problem for the elastic wave-based imaging of 3D anisotropic materials, where the order of magnitude of the number of the labels will be even larger than the presented 1600 labels of this paper.

FUTURE EXTENSION

The presented paper presents how the data can be generated and how the CNN is designed in a 2D setting, and the methodology is straightforward to follow. This research serves as the prototype in a 2D setting, and it can be extended into a 3D setting for detecting voids of arbitrary shapes and numbers in a 3D solid. Incorporation of the level-set method into the 3D SEM solver would be feasible. The explicit time integration (e.g., the Runge-Kutta scheme) coupled with the diagonal mass matrix that naturally arises (i.e., without the mass lumping approximation) in the 3D SEM will accelerate the generation of training data. The training of the CNN in the 3D setting should be performed in a GPU supercomputer while this 2D work was investigated using a workstation, with an NVIDIA Titan V GPU processor of 12 GB GPU memory, because the sizes of input and output data and their associated weight coefficient matrices are much larger in the 3D setting than the 2D counterpart.

We remark that real experimental data should be employed to validate the proposed elastodynamic element-wise classification method for identifying voids. However, in the presented study, we did not employ real experimental observational data because a recent work (Lloyd et al. 2023) indicated the difficulty of validating the inverse modeling in the 2D setting using real lab-scale experimental data. The previous work (Lloyd et al. 2023) presented an inverse-source modeling that was validated using experimental data of a very high-frequency range (e.g., 100 kHz) for a small aluminum block of 20 cm (L) $\times$ 15 cm (H) $\times$ 1.6 cm (D) as an experimental domain. It is the first of its kind that is aimed at reconstructing the actual force profile in space and time from a high-frequency actuator. The bottom of the block is attached to an experimentation table, but it was found that the ideal fixed-bottom boundary condition of the 2D FEM model hardly replicates the real bottom boundary in the experiment (i.e., the interface condition between an aluminum block and an experimentation table with a clamp and a soft cloth between the table and the block). The authors suggested waves that reflect off such a complex boundary and reach sensors may cause inaccuracies in the inversion result. Therefore, the source inversion was validated only in a limited setting where the source is located on the top surface of the aluminum block, and the observational duration is set to be sufficiently short so that the strong shear waves from the real bottom boundary do not reach the sensors, which were placed only on the top surface.

In other words, the experimental validation of the presented method necessitates the accurate experimental implementation of the ideal fixed bottom boundary in the 2D setting. Due to the aforementioned technical difficulty, it is not straightforward to experimentally validate the presented 2D study even though the presented 2D plane-strain setting is converted to the plane-stress one by changing the elasticity tensor. To bridge this gap, we suggest performing accurate modeling of the realistic bottom boundary condition in the experiment by using a high-performance 3D wave solver (e.g., the spectral element method (SEM) studied by Komatitsch and Tromp (1999) and Komatitsch et al. (2002)). The performance of such a potential neural network, for elementwise classification to identify voids in a 3D solid, could be validated using experimental data of a relatively long observation duration without excluding the reflected waves from the bottom boundary of a specimen.

Besides, we performed an iterative manual search for the selection of hyperparameters (such as the number of filters, 47, and filter sizes, 19) by iteratively examining the performance of CNN on the validation data set. We are aware that such an approach is time-consuming and laborious. Thus, we will investigate a new state-of-the-art neural ordinary differential equation (Neural ODE) (Chen et al. 2018) to remove this barrier of laborious hyperparameter search for the presented inverse-scattering problem.

DATA AVAILABILITY

Some or all data, models, or code generated or used during the study are available from the corresponding author by request.

- MATLAB code (.m format) of the presented forward modeling.
- MATLAB data sets (.mat format) of the presented numerical results.
- Tensorflow code (.py format) of the presented CNN modeling.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Award CMMI-2053694. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors are also grateful for the support by the Faculty Research and Creative Endeavors (FRCE) Research Grant-48058 at Central Michigan University. This paper is contribution # 181 of the Central Michigan University Institute for Great Lakes Research. The authors also greatly appreciate the reviewers' very constructive comments, which substantially helped improving the paper.

NOMENCLATURE

L, H	Horizontal and vertical lengths of the rectangular domain
x_0, y_0	The coordinates of the center of an ellipse
m, n	The major and minor axes of an ellipse
α	The angle between major axis and the x-axis of an ellipse
ν, ρ	Poisson's ratio and Mass density
E, ϵ	Young's modulus and Strain tensor
$V(x, y)$	Enrichment function
$\phi(x, y)$	Local shape function
$\mathbf{u}(x, y, t)$	Displacement field of a vector wave motion of a solid
$\boldsymbol{\sigma}(x, y, t)$	Cauchy stress tensor of a vector wave motion of a solid
v_p, v_s	Compressional wave velocity and shear wave velocity
$P(t)$	Ricker-pulse wave source signal over time t
f	Central frequency of the Ricker pulse
w	Weight coefficient between two consecutive neurons at adjacent layers
b	Previous layer feature or neuron
d	Size of the data from a previous layer
s	Bias associated with each neuron
O	Outcome after the application of weights and bias on CNN Layer
i	Subscript for the i-th element, i-th sample, and i-th neuron
j	Subscript for the j-th channel
k	Subscript for the k-th time step
f_{LReLU}	Leakly Rectified Linear Unit activation function
f_{sigmoid}	Sigmoid activation function
r	The activation function-applied outcome
a_i	The input of the activation function
N	Total number of elements
M	Total number of training data

$\mathcal{L}$	Loss function
A	Data matrix's non-normalized component
A^n	Data matrix's normalized component
$A_{\text{train}}^{\text{mean}}$	Mean value of the data matrix training set
$A_{\text{train}}^{\text{max}}$	Maximum value of the data matrix training set
$A_{\text{train}}^{\text{min}}$	Minimum value of the data matrix training set.

REFERENCES

Ashari, S. E. and Mohammadi, S. (2011). "Delamination analysis of composites by new orthotropic bimaterial extended finite element method." *International Journal for numerical methods in engineering*, 86(13), 1507–1543.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). "Yolov4: Optimal speed and accuracy of object detection." *arXiv preprint arXiv:2004.10934*.

Chatzi, E. N., Hiriyur, B., Waisman, H., and Smyth, A. W. (2011). "Experimental application and enhancement of the XFEM–GA algorithm for the detection of flaws in structures." *Computers & Structures*, 89(7), 556–570.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). "Neural ordinary differential equations." *Advances in neural information processing systems*, 31.

Elguedj, T., Gravouil, A., and Maigre, H. (2009). "An explicit dynamics extended finite element method. Part 1: mass lumping for arbitrary enrichment functions." *Computer Methods in Applied Mechanics and Engineering*, 198(30-32), 2297–2317.

Engwirda, D. (2005). "Unstructured mesh methods for the navier-stokes equations." Honors thesis, The University of Sydney, Sydney, Australia.

Engwirda, D. (2014). "Locally-optimal delaunay-refinement and optimisation-based mesh generation." Ph.D. thesis, The University of Sydney, Sydney, Australia.

Fathi, H., Vaez, S. H., Zhang, Q., and Alavi, A. H. (2021). "A new approach for crack detection in plate structures using an integrated extended finite element and enhanced vibrating particles system optimization methods." *Structures*, Vol. 29, Elsevier, 638–651.

Gao, Y., Liu, H., Wang, X., and Zhang, K. (2022). "On an artificial neural network for inverse scattering problems." *Journal of Computational Physics*, 448, 110771.

Glorot, X. and Bengio, Y. (2010). "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 249–256.

Guzina, B. and Pak, R. (1996). "Elastodynamic green's functions for a smoothly heterogeneous half-space." *International journal of solids and structures*, 33(7), 1005–1021.

Hellier, C. J. (2013). *Handbook of nondestructive evaluation*. McGraw-Hill Education.

Jeong, C., Na, S.-W., and Kallivokas, L. F. (2009). “Near-surface localization and shape identification of a scatterer embedded in a halfplane using scalar waves.” *Journal of Computational Acoustics*, 17(03), 277–308.

Jiang, S., Wan, C., Sun, L., and Du, C. (2022). “Flaw classification and detection in thin-plate structures based on sbfem and deep learning.” *International Journal for Numerical Methods in Engineering*.

Jiang, S., Zhao, L., and Du, C. (2021). “Combining dynamic XFEM with machine learning for detection of multiple flaws.” *International Journal for Numerical Methods in Engineering*, 122(21), 6253–6282.

Jung, J., Jeong, C., and Taciroglu, E. (2013). “Identification of a scatterer embedded in elastic heterogeneous media using dynamic XFEM.” *Computer Methods in Applied Mechanics and Engineering*, 259, 50–63.

Jung, J. and Taciroglu, E. (2014). “Modeling and identification of an arbitrarily shaped scatterer using dynamic XFEM with cubic splines.” *Computer Methods in Applied Mechanics and Engineering*, 278, 101–118.

Jung, J. and Taciroglu, E. (2016). “A divide-alternate-and-conquer approach for localization and shape identification of multiple scatterers in heterogeneous media using dynamic XFEM.” *Inverse Problems & Imaging*, 10(1), 165.

Khatir, S. and Wahab, M. A. (2019). “A computational approach for crack identification in plate structures using XFEM, XIGA, PSO and Jaya algorithm.” *Theoretical and Applied Fracture Mechanics*, 103, 102240.

Komatitsch, D., Ritsema, J., and Tromp, J. (2002). “The spectral-element method, Beowulf computing, and global seismology.” *Science*, 298(5599), 1737–1742.

Komatitsch, D. and Tromp, J. (1999). “Introduction to the spectral element method for three-dimensional seismic wave propagation.” *Geophysical Journal International*, 139(3), 806–822.

Livani, M., Khaji, N., and Zakian, P. (2018). “Identification of multiple flaws in 2D structures using dynamic extended spectral finite element method with a universally enhanced meta-heuristic optimizer.” *Structural and multidisciplinary optimization*, 57(2), 605–623.

Lloyd, S., Schaal, C., and Jeong, C. (2023). “Inverse modeling and experimental validation for reconstructing wave sources on a 2D solid from surficial measurement." *Ultrasonics*, 128, 106880.

Ma, C., Yu, T., Thanh-Tung, N., Bui, T. Q., et al. (2020). "Detection of multiple complicated flaw clusters by dynamic variable-node XFEM with a three-step detection algorithm." *European Journal of Mechanics-A/Solids*, 82, 103980.

Maharjan, S., Guidio, B., Fathi, A., and Jeong, C. (2022). "Deep and convolutional neural networks for identifying vertically-propagating incoming seismic wave motion into a heterogeneous, damped soil column." *Soil Dynamics and Earthquake Engineering*, 162, 107510.

Menouillard, T., Song, J.-H., Duan, Q., and Belytschko, T. (2010). "Time dependent crack tip enrichment for dynamic crack propagation." *International Journal of Fracture*, 162(1-2), 33–49.

Nanthakumar, S., Lahmer, T., and Rabczuk, T. (2013). "Detection of flaws in piezoelectric structures using extended FEM." *International Journal for Numerical Methods in Engineering*, 96(6), 373–389.

Newmark, N. M. (1959). "A method of computation for structural dynamics." *Journal of the engineering mechanics division*, 85(3), 67–94.

Rabinovich, D., Givoli, D., and Vigdergauz, S. (2007). "XFEM-based crack detection scheme using a genetic algorithm." *International Journal for Numerical Methods in Engineering*, 71(9), 1051–1080.

Rabinovich, D., Givoli, D., and Vigdergauz, S. (2009). "Crack identification by 'arrival time' using XFEM and a genetic algorithm." *International Journal for Numerical Methods in Engineering*, 77(3), 337–359.

Stanchits, S., Burghardt, J., and Surdi, A. (2015). "Hydraulic fracturing of heterogeneous rock monitored by acoustic emission." *Rock Mechanics and Rock Engineering*, 48(6), 2513–2527.

Sukumar, N., Huang, Z., Prévost, J.-H., and Suo, Z. (2004). "Partition of unity enrichment for bimaterial interface cracks." *International journal for numerical methods in engineering*, 59(8), 1075–1102.

Sun, H., Waisman, H., and Betti, R. (2013). "Nondestructive identification of multiple flaws using XFEM and a topologically adapting artificial bee colony algorithm." *International Journal for Numerical Methods in Engineering*, 95(10), 871–900.

Sun, H., Waisman, H., and Betti, R. (2014). "A multiscale flaw detection algorithm based on

XFEM." *International Journal for Numerical Methods in Engineering*, 100(7), 477–503.

Taylor, R. (2017). *FEAPpv – A finite element analysis program. Personal version 4.1 user manual.* University of California at Berkeley.

Waisman, H., Chatzi, E., and Smyth, A. W. (2010). "Detection and quantification of flaws in structures by the extended finite element method and genetic algorithms." *International Journal for Numerical Methods in Engineering*, 82(3), 303–328.

Wang, Y. and Waisman, H. (2017). "Material-dependent crack-tip enrichment functions in xfem for modeling interfacial cracks in bimaterials." *International Journal for Numerical Methods in Engineering*, 112(11), 1495–1518.

Wrobel, L. C. (2002). *The boundary element method, volume 1: Applications in thermo-fluids and acoustics*, Vol. 1. John Wiley & Sons.

Yan, G., Sun, H., and Waisman, H. (2015). "A guided bayesian inference approach for detection of multiple flaws in structures using the extended finite element method." *Computers & Structures*, 152, 27–44.

Zhang, C., Wang, C., Lahmer, T., He, P., and Rabczuk, T. (2016). "A dynamic XFEM formulation for crack identification." *International Journal of Mechanics and Materials in Design*, 12(4), 427–448.

Zhang, L., Yang, G., Hu, D., and Han, X. (2019). "An approach based on level set method for void identification of continuum structure with time-domain dynamic response." *Applied Mathematical Modelling*, 75, 446–480.

List of Tables

TABLE 1. The evaluation metric for CNN for the test data set,

Figures	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Fig. 14 - Best CNN Prediction	100.00	100.00	100.00	100.00
Fig. 15 - Worst CNN Prediction	99.97	99.18	99.90	99.54

TABLE 2. The evaluation metrics – accuracy, precision, recall, and f1-score – for CNN for the test data set from an independent FEM wave solver that does not use level-set-method,

Figures	Type of targeted voids	Accuracy	Precision	Recall	F1-score
Fig. 16b	Two horizontally-aligned ellipses (Data without circles)	98.62	34.38	91.67	50.00
Fig. 16c	Two horizontally-aligned ellipses (Data with circles)	99.31	71.88	92.00	80.70
Fig. 17b	Two vertically-aligned ellipses (Data without circles)	98.50	28.12	90.00	42.86
Fig. 17c	Two vertically-aligned ellipses (Data with circles)	99.19	62.50	95.24	75.47
Fig. 18b	Three elliptical voids	95.94	25.00	37.84	30.11
Fig. 18c	Three elliptical voids (Data with circles)	93.56	48.21	26.73	34.39
Fig. 19b	One circular void	95.94	18.75	81.82	30.51
Fig. 19c	One circular void (Data with circles)	99.44	88.75	100.00	94.04
Fig. 20b	Two vertically-aligned circular voids	92.06	27.04	79.63	40.38
Fig. 20c	Two vertically-aligned circular voids (Data with circles)	97.38	75.47	97.56	85.11

List of Figures

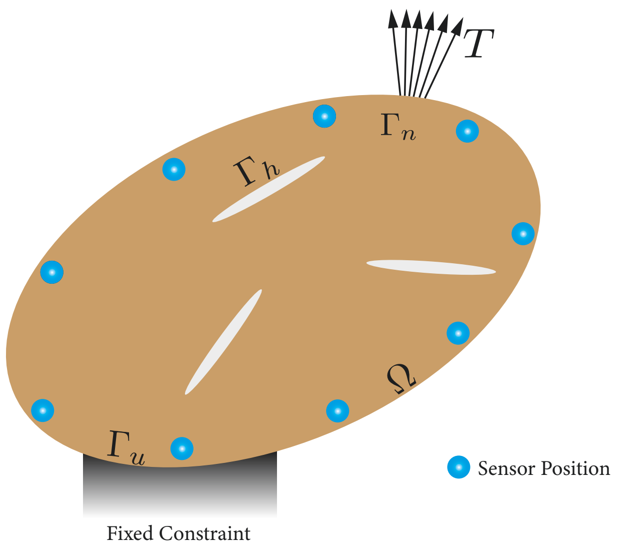


Fig. 1. Boundary value problem for an elastic medium with voids.

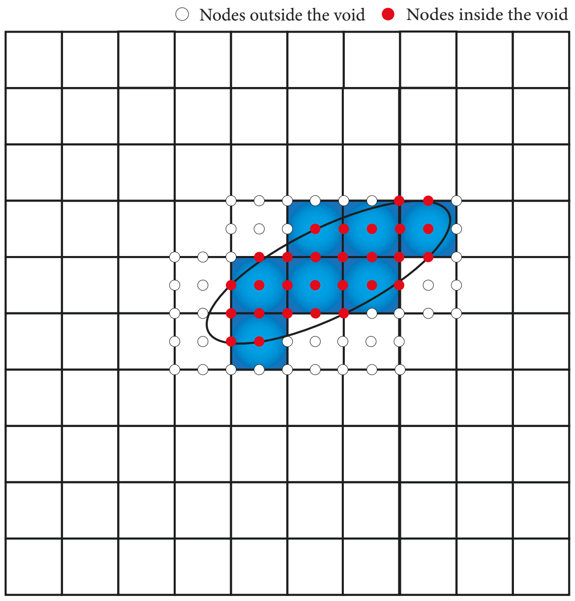
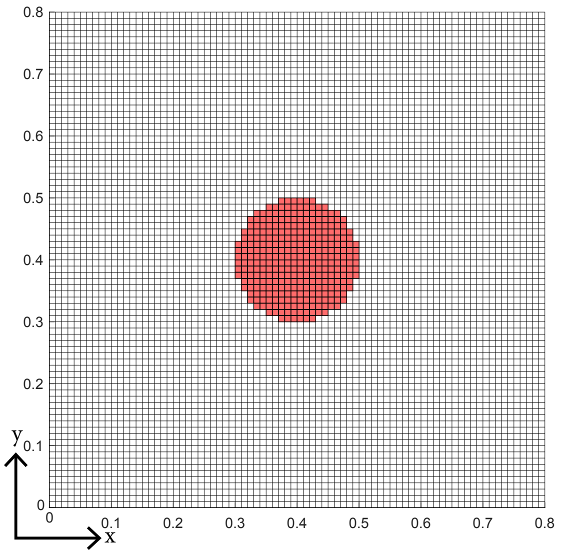
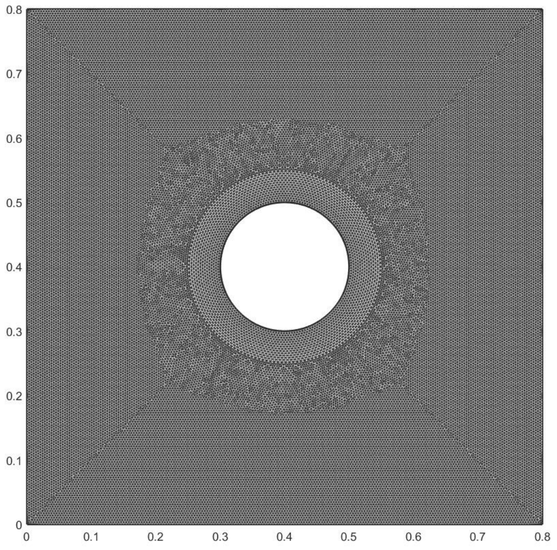


Fig. 2. Level-set categorization of an element into a void and a non-void element for a 9-node element: the red/white dots indicate the nodes inside/outside a void. If the centroid and 4 other nodes (out of 9 nodes) of an element are inside the boundary of an actual void, that element is considered a void element. Otherwise, it is defined as a non-void element.

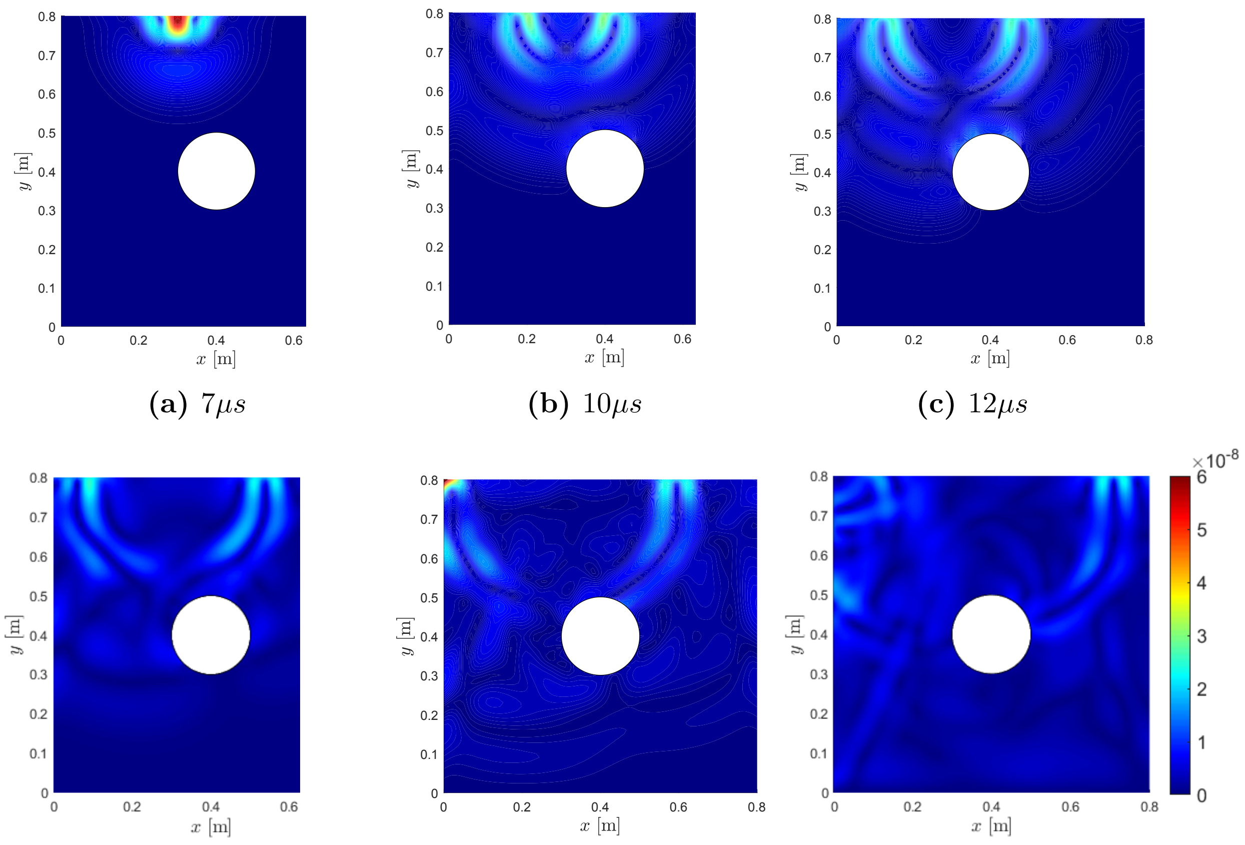


(a)



(b)

Fig. 3. (a) The structured background mesh for our level-set forward wave solver (the elements in red represent those that are considered voids) and (b) the unstructured mesh for the Finite Element Analysis Program (FEAP), which explicitly models the boundary of a void.



(a) $7\mu s$ **(b)** $10\mu s$ **(c)** $12\mu s$

(d) $14\mu s$ **(e)** $16\mu s$ **(f)** $20\mu s$

Fig. 4. Contour plot showing the amplitudes of the displacement field of elastic waves in an isotropic homogeneous aluminum of a plane-strain setting with a void at (a) 7μs, (b) 10 μs, (c) 12 μs, (d) 14 μs, (e) 16 μs, and (f) 20 μs.

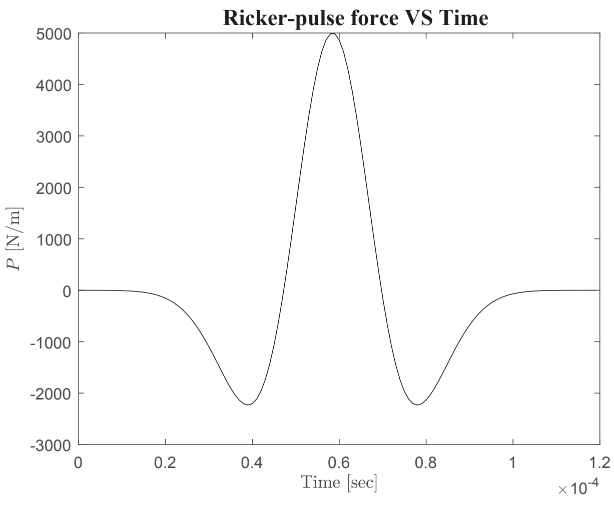


Fig. 5. A plot showing the Ricker-pulse wave signal.

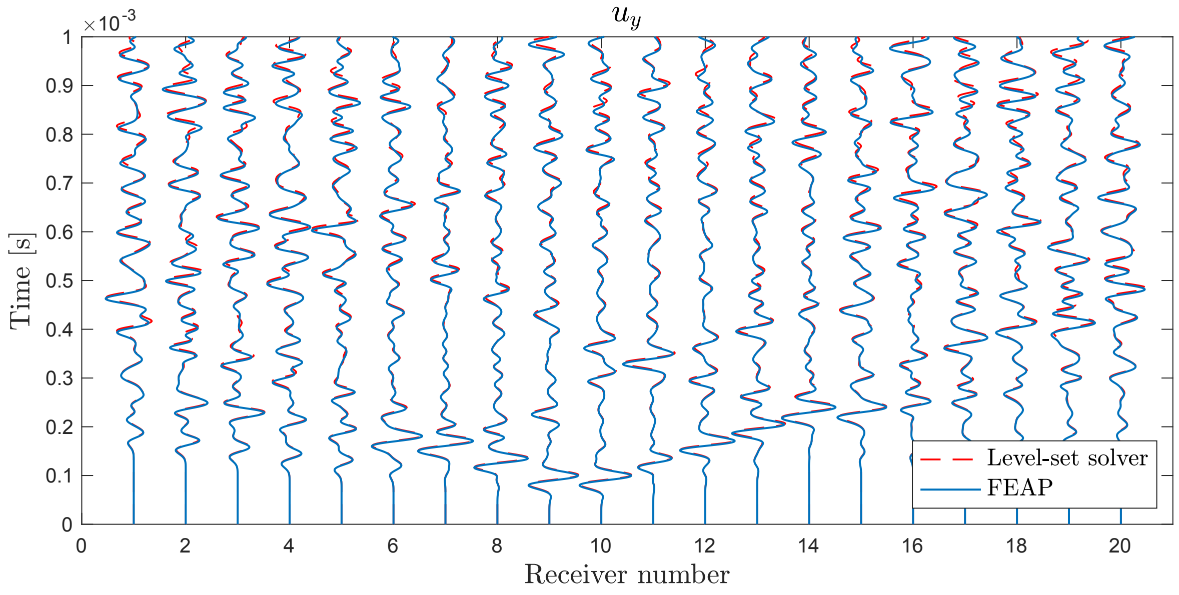


Fig. 6. Comparison between u_y generated by the presented level-set wave solver versus FEAP at 20 different locations.

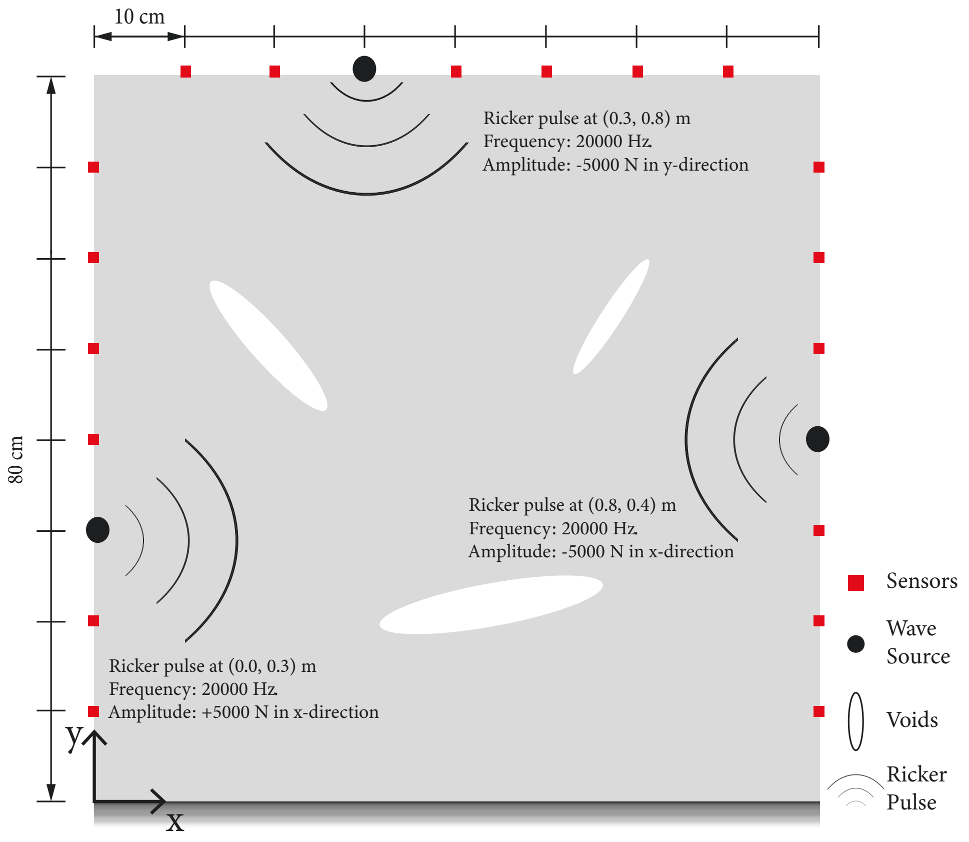
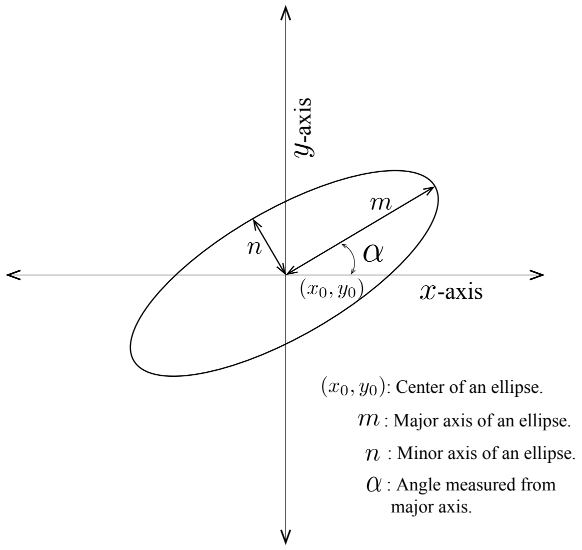
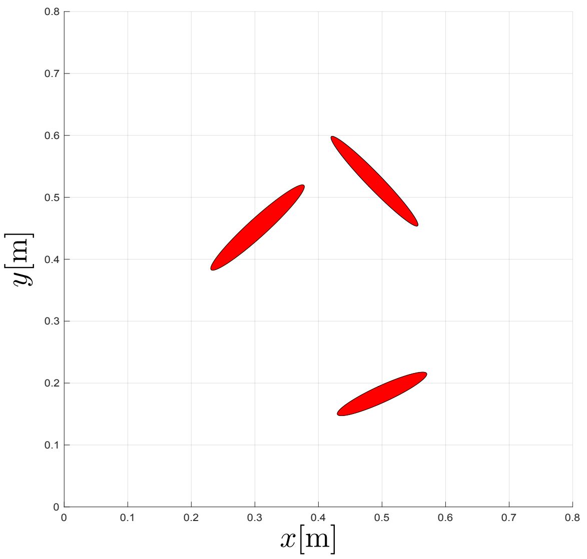

10 cm
80 cm
Ricker pulse at (0.3, 0.8) m
Frequency: 20000 Hz.
Amplitude: -5000 N in y-direction
Ricker pulse at (0.8, 0.4) m
Frequency: 20000 Hz.
Amplitude: -5000 N in x-direction
Ricker pulse at (0.0, 0.3) m
Frequency: 20000 Hz.
Amplitude: +5000 N in x-direction
y
x
Sensors
Wave Source
Voids
Ricker Pulse

Fig. 7. A schematic of multiple voids inside a domain with sensors and wave sources for the data generation.



y-axis
m
n
α
(x_0, y_0)
x-axis
(x_0, y_0): Center of an ellipse.
m : Major axis of an ellipse.
n : Minor axis of an ellipse.
α : Angle measured from major axis.

(a) The parameters of an ellipse used for random data generation.



(b) A sample of random number of voids, generated using our randomizer.

Fig. 8. Generation of elliptical voids in data sets.

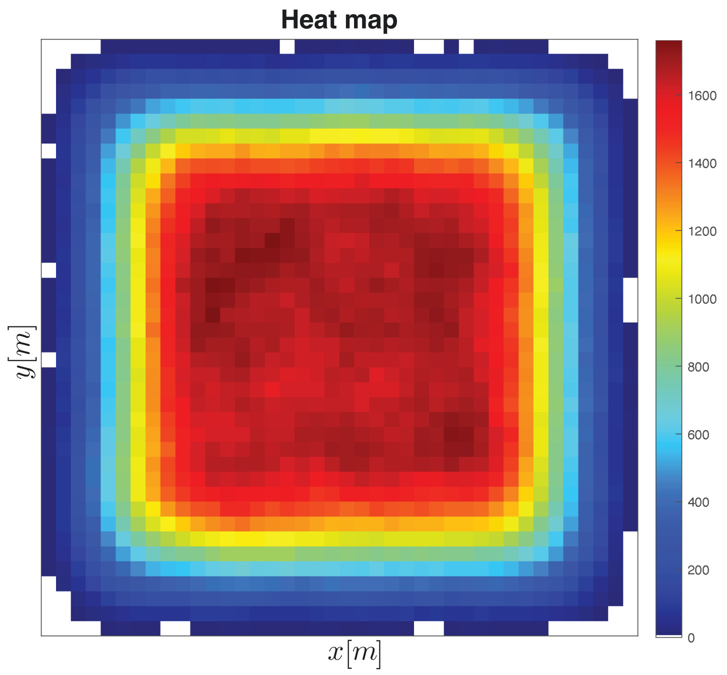
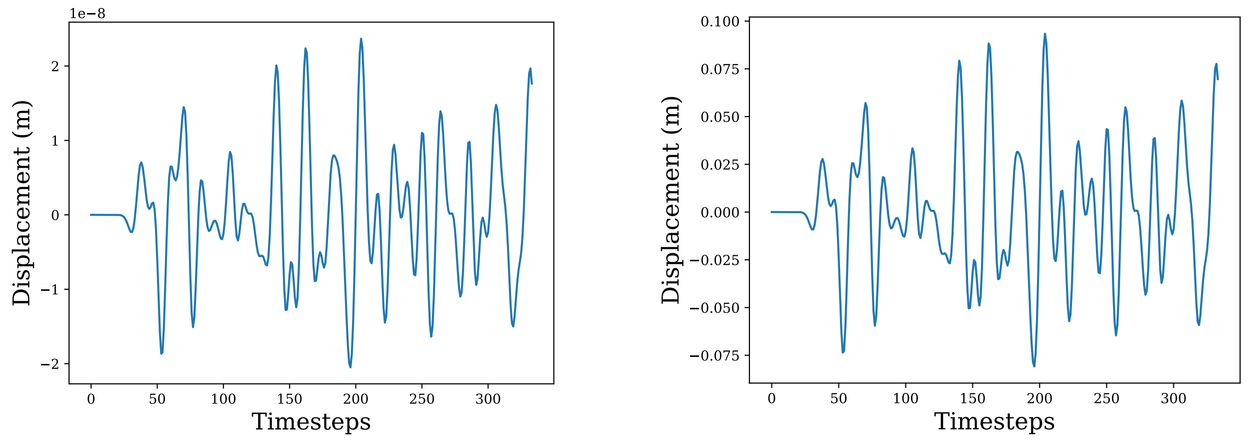


Fig. 9. This heat map shows, over the entire 40×40 element grid of the domain, how many times a given element is recognized as a void by the level set approximation during the generation of 36000 data sets.



(a) Before normalization. **(b)** After normalization.

Fig. 10. Input-layer displacement waveforms in training data.

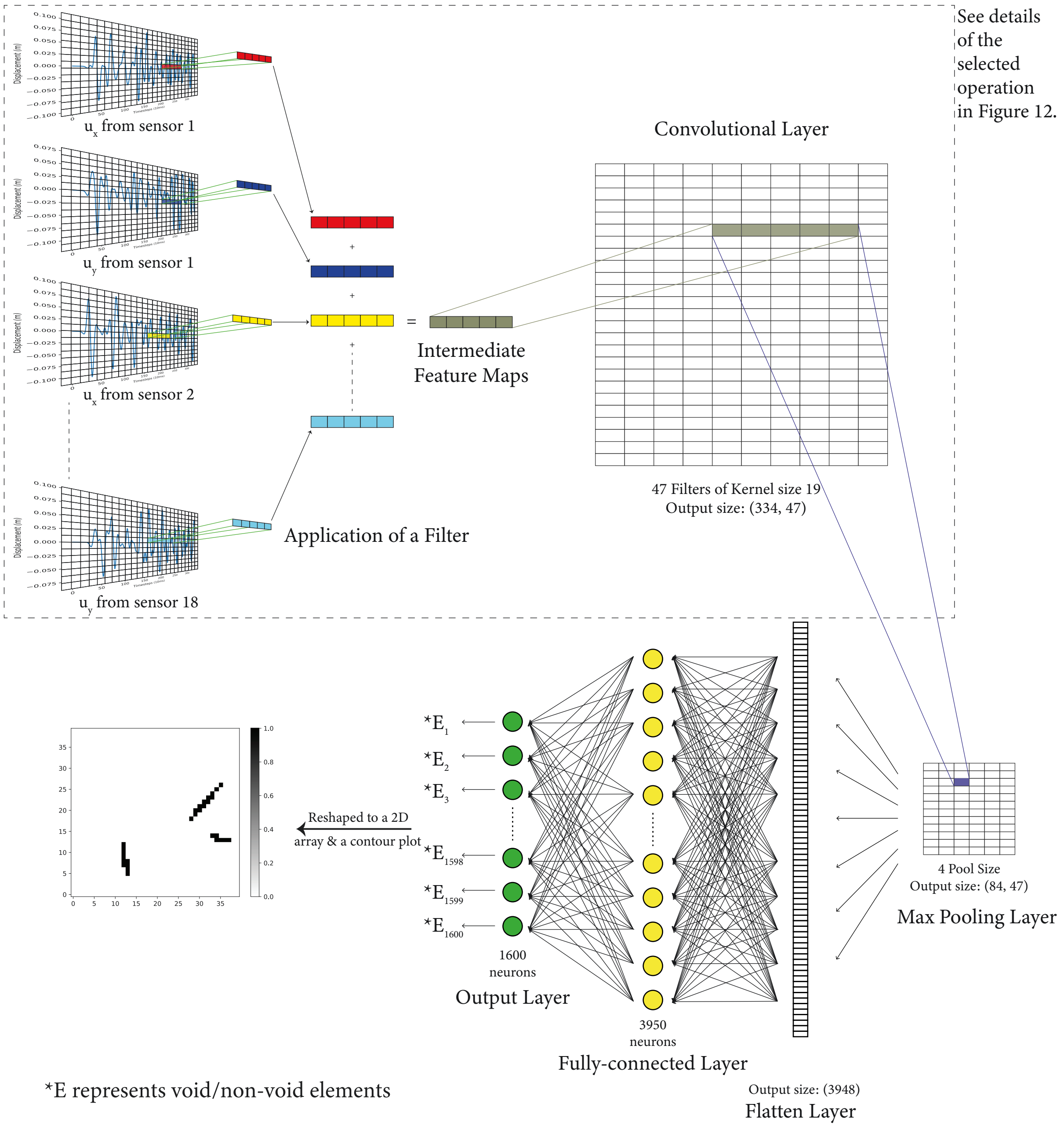


Fig. 11. The architecture of our CNN, which relates measured wave signals to element types.

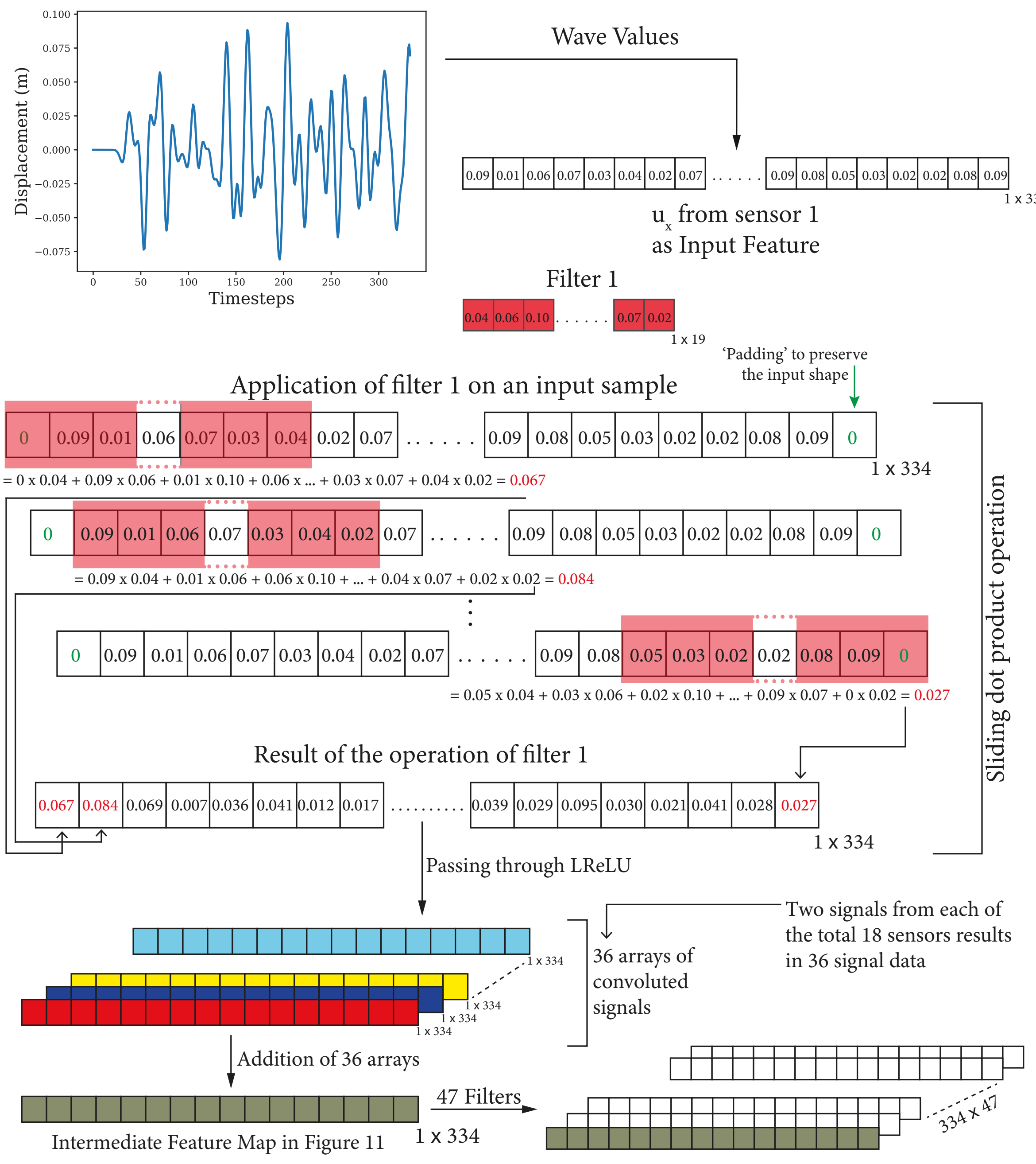
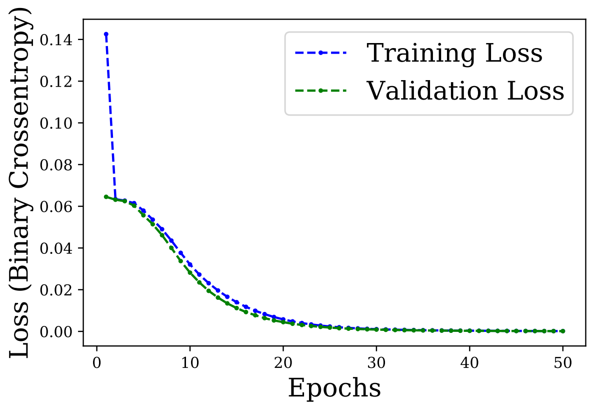
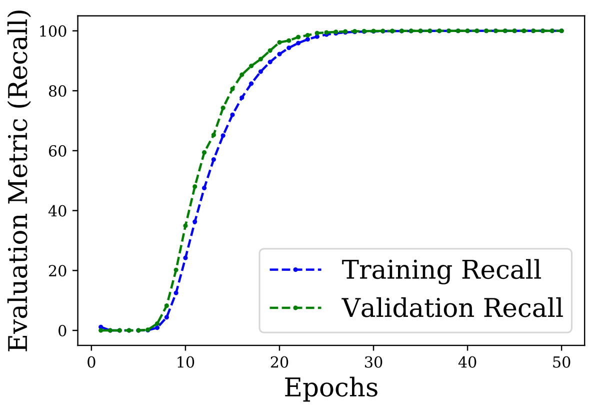

Displacement (m)
Timesteps
Wave Values
u_x from sensor 1 as Input Feature
1 x 334
Filter 1
1 x 19
Application of filter 1 on an input sample
'Padding' to preserve the input shape
= 0 x 0.04 + 0.09 x 0.06 + 0.01 x 0.10 + 0.06 x ... + 0.03 x 0.07 + 0.04 x 0.02 = 0.067
= 0.09 x 0.04 + 0.01 x 0.06 + 0.06 x 0.10 + ... + 0.04 x 0.07 + 0.02 x 0.02 = 0.084
= 0.05 x 0.04 + 0.03 x 0.06 + 0.02 x 0.10 + ... + 0.09 x 0.07 + 0 x 0.02 = 0.027
Sliding dot product operation
Result of the operation of filter 1
Passing through LReLU
36 arrays of convoluted signals
Two signals from each of the total 18 sensors results in 36 signal data
Addition of 36 arrays
Intermediate Feature Map in Figure 11
47 Filters
334 x 47

Fig. 12. Application of convolutional layer filters on the input data set (the beginning part of Figure 11). The numbers in the figure do not represent actual feature input or filter weights.

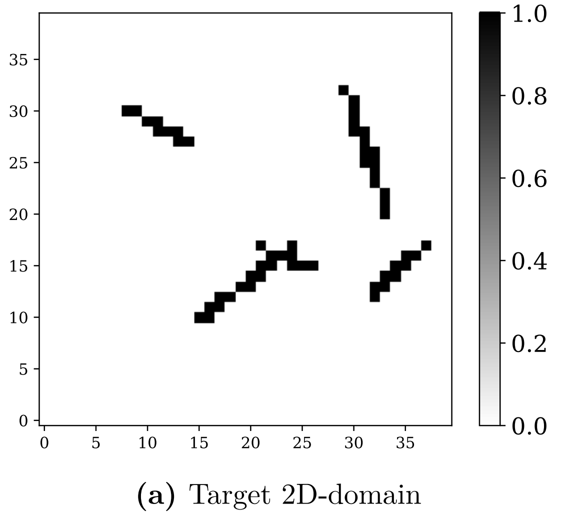


(a) Convergence of the loss during training for 50 epochs.

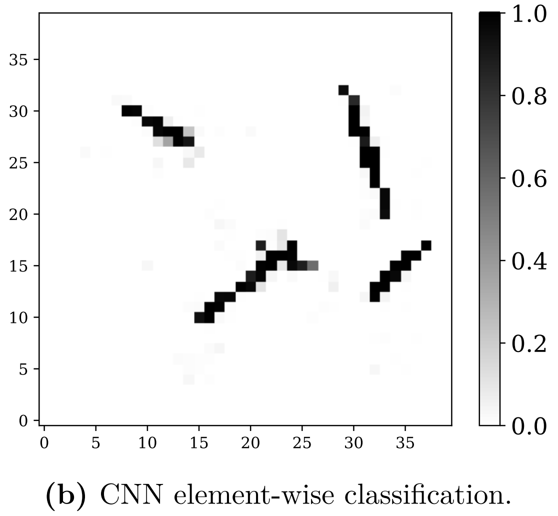


(b) Increase in recall metric during training for 50 epochs.

Fig. 13. Convergence of loss and increase in evaluation metric (recall) for both training and validation data set for 50 epochs.

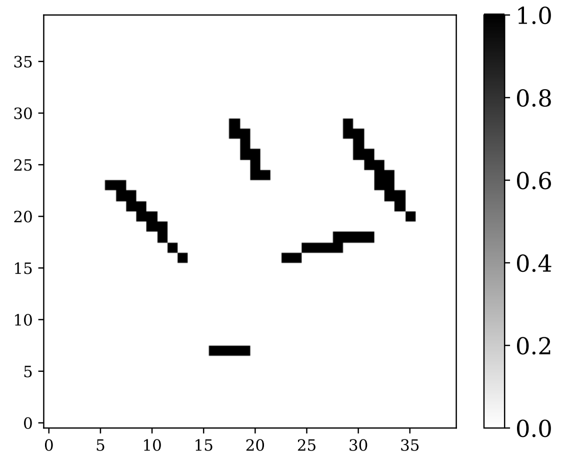


(a) Target 2D-domain

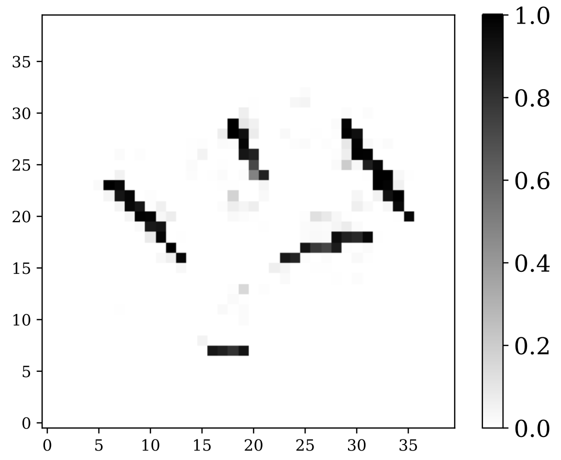


(b) CNN element-wise classification.

Fig. 14. Best void prediction from a test data set, which is made using 5 elliptical voids.

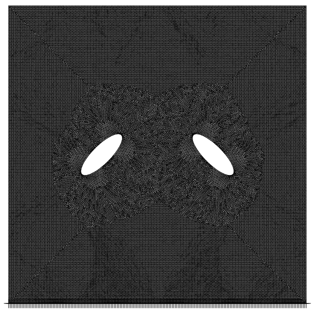


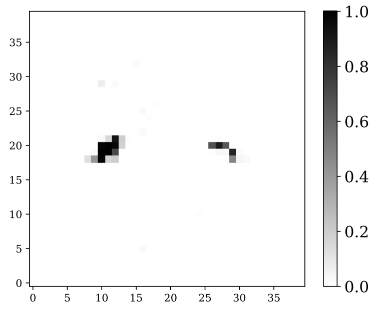
(a) Target 2D-domain



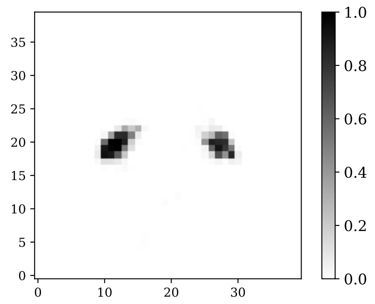
(b) CNN element-wise classification.

Fig. 15. Worst void prediction from a test data set, which is made using 5 elliptical voids.


(a) 2 target elliptical voids modeled by FEAP.

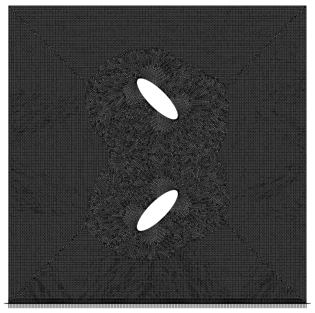


(b) CNN element-wise classification

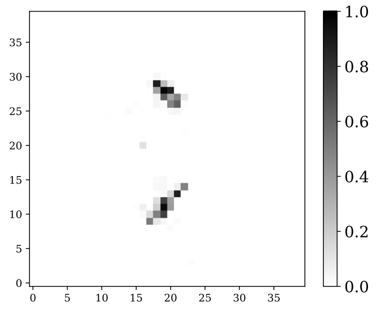


(c) CNN element-wise classification (trained with circles).

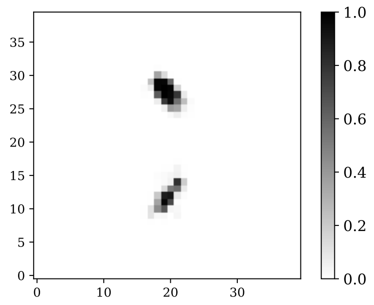
Fig. 16. Detection of 2 horizontally-aligned elliptical voids from blind test measurement data by our CNN.



(a) 2 target elliptical voids modeled by FEAP.

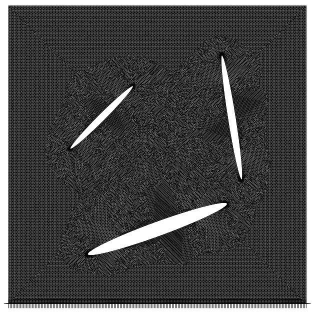


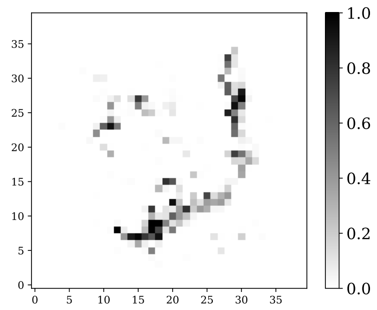
(b) CNN element-wise classification.



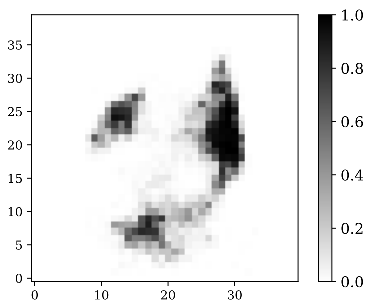
(c) CNN element-wise classification (trained with circles).

Fig. 17. Detection of 2 vertically-aligned elliptical voids from blind test measurement data by our CNN.


(a) 3 target elliptical voids modeled by FEAP.

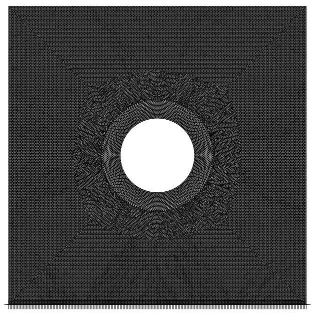


(b) CNN element-wise classification.

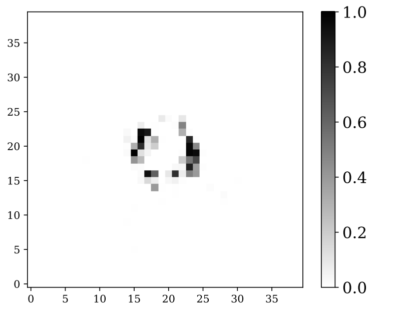


(c) CNN element-wise classification (trained with circles).

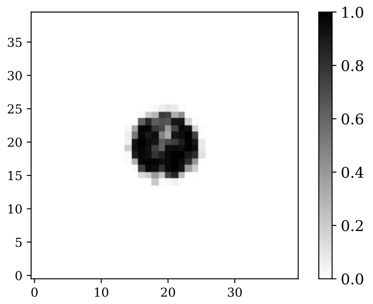
Fig. 18. Detection of 3 elliptical voids from blind test measurement data by our CNN.



(a) 1 target circular void modeled by FEAP.

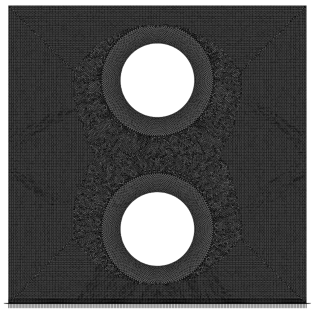


(b) CNN element-wise classification.

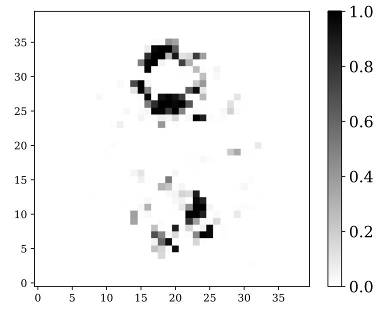


(c) CNN element-wise classification (trained with circles).

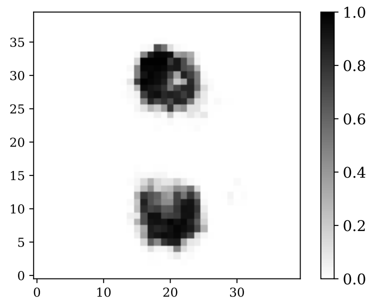
Fig. 19. Detection of 1 circular void from blind test measurement data by our CNN.



(a) 2 target circular voids modeled by FEAP.



(b) CNN element-wise classification.



(c) CNN element-wise classification (trained with circles).

Fig. 20. Detection of 2 vertically aligned-circular voids from blind test measurement data by our CNN.