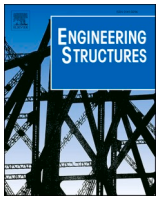
Engineering Structures 330 (2025) 119884

Contents lists available at ScienceDirect

Engineering Structures

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



# Automated machine learning exact dirichlet boundary physics-informed neural networks for solid mechanics

Xiaoge Tian [a], Jiaji Wang [a,*], Chul-Woo Kim [b], Xiaowei Deng [a], Yingjie Zhu [c]

[a] Department of Civil Engineering, The University of Hong Kong, China
[b] Department of Civil and Earth Resources Engineering, Kyoto University, Kyoto, Japan
[c] School of Civil Engineering, North China University of Technology, China

ARTICLE INFO

*Keywords:*
Solid mechanics
Physics-informed neural network (PINN)
Exact Dirichlet boundary PINN (EPINN)
Bayesian-optimization tree-structured parzen estimator
AutoML EPINN (AEPINN)

ABSTRACT

While Physics-informed neural networks (PINN) have made significant progress in solving partial differential equations (PDE), conventional PINN may have convergence issues due to spectral bias, the requirement of loss balancing, and a significant number of trainable weights. Exact Dirichlet boundary condition Physics-informed Neural Networks (EPINN) was developed to solve forward problems in solid mechanics by applying tensor decomposition, approximating distance function, and the principle of least work, achieving more than 127 times speedup compared to PINN. However, the sensitivity of hyperparameters of the PINN framework is less reported. To merge the gap, this study develops the mesh-free 3D Bayesian-Optimization Tree-Structured Parzen Estimator (BO-TPE) Automated Machine Learning EPINN to solve solid mechanics problems without labelled data of the solution field. Developed based on Nvidia modulus platform, the Automated Machine Learning EPINN (AEPINN) can achieve more than 20 times speedup for 2D plane stress problems and four times speedup for 3D bracket problems compared with the EPINN architecture. Compared with conventional PINN, AEPINN model achieved more than 200 times speedup for a plane stress problem and 400 times speed up for a bracket problem. For a two-span three-story frame composed of beams, columns, and slabs, the AEPINN model can simulate the frame displacement deformations comparable to ABAQUS results with adequate accuracy and speed with GPU accelerated. Optimized hyperparameters AEPINN can approach a hyperelastic cube rubber case within 60 s compared with Abaqus results of Mooney-Rivlin constitutive law. The comparison between single-precision and double-precision training is also illustrated. The influences of hyperparameters in the adopted EPINN framework are examined accordingly.

## 1. Introduction

As one of the numerical solution methods to obtain approximated solution field Partial Differential Equations (PDE), Finite Element Methods (FEM) can mesh complex objects into simple elements achieving numerical results approximation [1]. Despite the performance of FEM, it may be hard to obtain the derivatives of the solution field with respect to input parameters, especially for complicated structures. Hence, solving parametric simulation problems, inverse problems and design optimization problems may be challenging due to the numerical approximation in FEM [2,3]. GPU-accelerated differentiable solvers for solid mechanics problems have been rapidly developed to achieve significant speedup in parametric analysis, design optimization problems, and inverse problems. Deep learning has recently contributed to intelligent computation research with substantial efficiencies and affordable costs in simulation and experimental parts [4–7]. Researchers added loss terms of governing equations (PDE loss) to the data loss terms and developed a physics-informed neural network framework (PINN) to solve scientific PDE computation problems by utilizing the Universal Approximation Theory of Deep Neural Network (DNN) [8–11]. Compared with DNN solvers, PINN can calculate fluid mechanics and solid mechanics problems with acceptable errors without labeled data of the solution field [12]. While the efficiency may be lower than numerical solvers, PINN has been developed to address simple fluid, 1D, and 2D static solid mechanical problems [13–15]. However, the extension of the PINN framework for solving complex solid mechanics problems is rarely reported due to the lack of efficient PINN-based solvers.

Hierarchical Deep-learning Neural Network (HiDeNN) framework was developed to reformulate the global shape function of finite

* Corresponding author.
*E-mail address:* cewang@hku.hk (J. Wang).

https://doi.org/10.1016/j.engstruct.2025.119884
Received 10 September 2024; Received in revised form 17 January 2025; Accepted 3 February 2025
Available online 19 February 2025

**NOMENCLATURE**

| | |
|---|---|
| $x_I$ | The coordinate of the node $I$ |
| $N_I(x)$ | The shape function of node $I$ |
| $N(x)$ | The global shape function |
| $u_I$ | The predicted displacement of node $I$ |
| $u(x)$ | The global displacement function |
| $Q$ | Tensor decomposition modes |
| $\beta_I^{(q)}$ | The $q$th trainable weights of the tensor decomposition in $x$ direction |
| $\gamma_J^{(q)}$ | The $q$th trainable weights of the tensor decomposition in y direction |
| $\theta_K^{(q)}$ | The $q$th trainable weights of the tensor decomposition in $z$-direction |
| $\mathrm{u}_{\mathrm{TD}}$ | Tensor decomposition predicted displacement |
| $EI_{y^*}(x)$ | Expected improvement |
| $l(x)$ | Density functions of the bad performances |
| $g(x)$ | Density functions of good performances |
| $\mathscr{L}_{energy}(\mathrm{u};\mathrm{f},\bar{\mathrm{t}})$ | Energy loss function |
| $\mathrm{u}(u,v,w)$ | Final predicted displacement field |
| $\widehat{W}$ | Trainable weights |
| $O$ | Automated machine learning objective function of the AEPINN model |

elements by inputting nodal coordinates [16]. Weights and biases of HiDeNN were theoretically derived from the shape function of finite elements to reformulate the spatial discretization of finite elements [16]. HiDeNN significantly reduced the number of trainable variables and improved the efficiency and ability of HiDeNN in analyzing PDE problems [17–19]. Exact Dirichlet boundary condition PINN (EPINN) framework was developed [20] by adopting the Approximate Distance Function (ADF) to enforce the Dirichlet boundary conditions exactly so that the boundary conditions can be comprehensively satisfied [21]. Compared with conventional PINN, EPINN can efficiently embed exact Dirichlet boundary conditions, which are regulated by the principle of least work. To further improve efficiency, EPINN applies tensor decomposition in solving complicated structures to reduce the training time and improve accuracy. The results show that EPINN can be applied to solve 3D solid mechanics problems. Although EPINN is an efficient differentiable solver for solid mechanics problems, reasonable input of hyperparameters may still affect the performance of EPINN [22]. For most deep learning architectures, the learning rate and rate decay steps will influence the speed of updating the trainable parameters. For EPINN, although the network architecture does not allow modification to the number of layers, the modes of Tensor Decomposition (TD) can affect the nonlinear fitting ability.

However, there remain many parameters with uncountable sets for users to pick before testing problems, which would be very inefficient if the proposed method is highly sensitive to the machine learning components. Researchers, therefore, are adopting processes to make machines learn those unknown effects automatically. Conventional Automated machine learning consists of data preparation, feature engineering, architecture design and hyperparameter optimization. For physics-driven frameworks, input information commonly includes coordinates and parameters, where data preparation and feature engineering workloads are normally fixed with several effective methods [23]. For PINNs, network architecture for advanced versions normally are designed to be effective for the studied type of framework. Hence, hyperparameters are normally adjustable for most physics-informed machine-learning tasks. For physics-informed machine learning frameworks, especially for estimating unknown surrogate models or functions in forward or inverse problems, sufficient hyperparameters can contribute to problem convergence and avoid overfitting issues [12,23,24]. Exploring the possible sets of multiple hyperparameters is pretty time-consuming and it is difficult to directly handle these black boxes by Automated Machine Learning (AutoML) without appropriate methods [25]. To obtain information on the required objective function, Bayesian Optimization (BO), as one of the powerful algorithms in improving efficiencies during searching feasible space, has become very effective in accelerating the hyperparameters estimation scenarios by adopting the Bayesian theorem to update the surrogate model of the objective function with observed results[26]. Approximating surrogate models, BO will calculate the expected improvement of the next selected set to ensure the sampling of hyperparameter sets is contributing to the reduction of optimization iterations significantly [26,27]. Among various BO framework structured cases, Tree-Structured Parzen Estimator (TPE) separates selection performance into good or bad sets while considering the prior experiences and adjusting the search strategy based on this[28,29]. In most cases, TPE can solve parameter tuning problems efficiently, especially for searching optimized hyperparameter sets in high-dimensional space.

In this study, we developed the EPINN with Bayesian Optimization Tree-Structured Parzen Estimator (BO-TPE)[30,31]as the optimization module for automated machine learning EPINN (AEPINN). Fig. 1 states the developed AEPINN architecture. Compared with current PINN studies, our contributions are as follows: we develop the 3D irregularly shaped mesh-free artificial intelligence framework in addressing the solid mechanics linear elastic problems with 3D file format STL as input, which can be obtained through multiple methods, indicating that the proposed framework can be adopted to solve any shaped mechanics problems in the future. Secondly, the automated machine learning framework for physics-informed machine learning-based structures is established and can be extended to the current PINN framework for accelerating scientific computing. The obtained optimized hyperparameters can be directly utilized in other solid mechanics problems to achieve a speedup and improve accuracy without rerunning the automated process for every problem, improving the AutoML efficiency. Thirdly, we discover hyperparameter balancing can be critical for reducing the searching space to further accelerate the process of solving solid mechanics problems. With balanced searching space, automated machine learning time can be reduced ten times compared with directly applying AutoML. Last but not least, with optimized hyperparameters this study tackles the 3D complex shaped frame problems and 3D hyperelastic cube problems efficiently within 100 seconds, exhibiting the potential of artificial intelligence in addressing large-scale and nonlinear problems in the future.

Fig. 1 illustrates the physics-informed 3D scanned objective solid mechanics problems-solving framework based on AEPINN. The 3D model can be formulated through designing or analyzing software such as Autodesk Computer Aided Design (AutoCAD), Blender software or 3D scanners to obtain StereoLithography (STL), Odb, and any other object files contain point cloud information for mechanics analysis. Similar to FEM models, different meshing grids may influence the numerical analysis accuracy and efficiency respectively, meaning the hyperparameters in shape functions have to be analyzed accordingly based on the specific case. Therefore, this study proposes an AEPINN architecture-based automated machine learning methodology to improve the computational ability of artificial intelligence. Continuous Section 2 mainly illustrates the methodology of the AEPINN framework. Section 3 tests the performance of the AEPINN framework in solving 2D plane stress, 3D bracket traction, and 3D complex-shaped frame structure. Section 4 concludes the overall performance of the AEPINN framework for the aforementioned demonstrations.

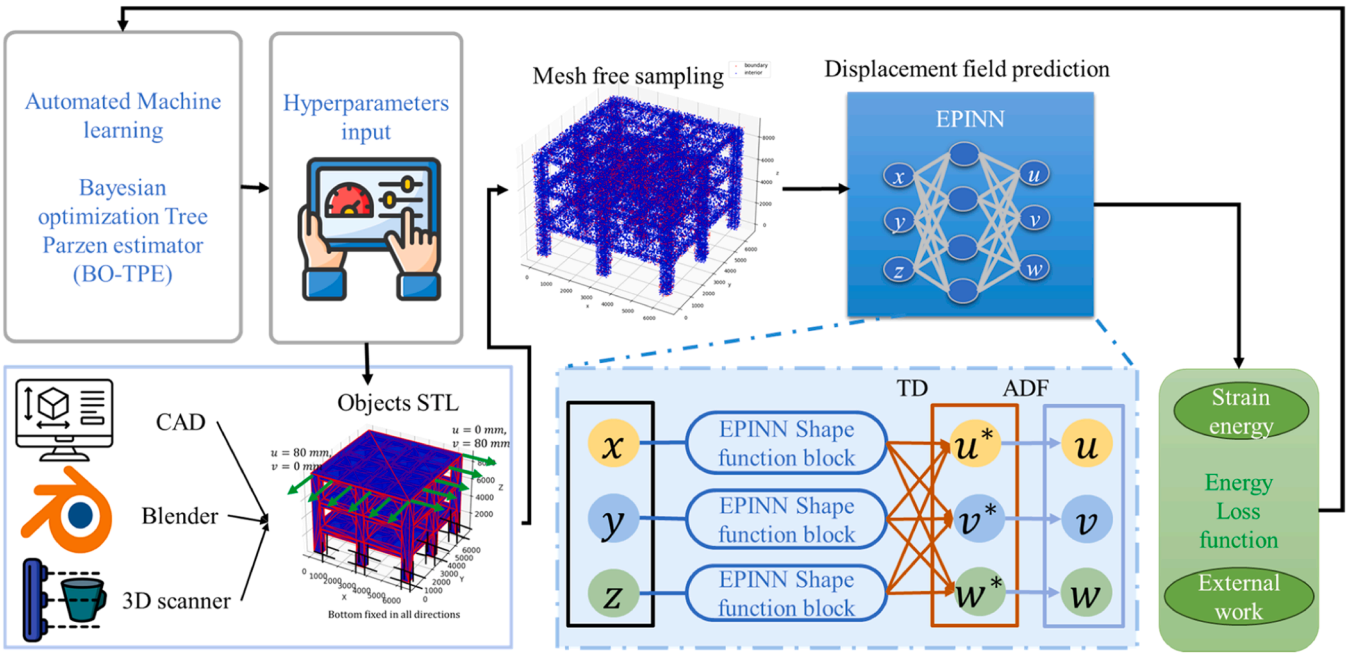



**Fig. 1.** The framework of the proposed AEPINN research.

## 2. Model architecture

### 2.1. The model architecture of AEPINN

Fig. 2 illustrates the overall framework of this research, the 3D STL file will first be sampled into thousands of points before inputting for AEPINN analysis. Different from the conventional PINN, this framework is purely mesh-free so that there will be no mesh grids for the three-dimensional objects, which can be applied to any irregularly shaped objects compared with normal meshing-based physics-informed methodologies. Before the optimization process, the ranges of hyperparameters should be defined based on preliminary results. Therefore, how to balance different hyperparameters ranges is of great significance for the successful automated machine learning process. Hence, the hyperparameters balance will first be conducted to ensure the reasonable range for the continuous automated machine learning which is of great importance for reducing the computational costs as well as improving the efficiency of appropriate hyperparameters for solving the

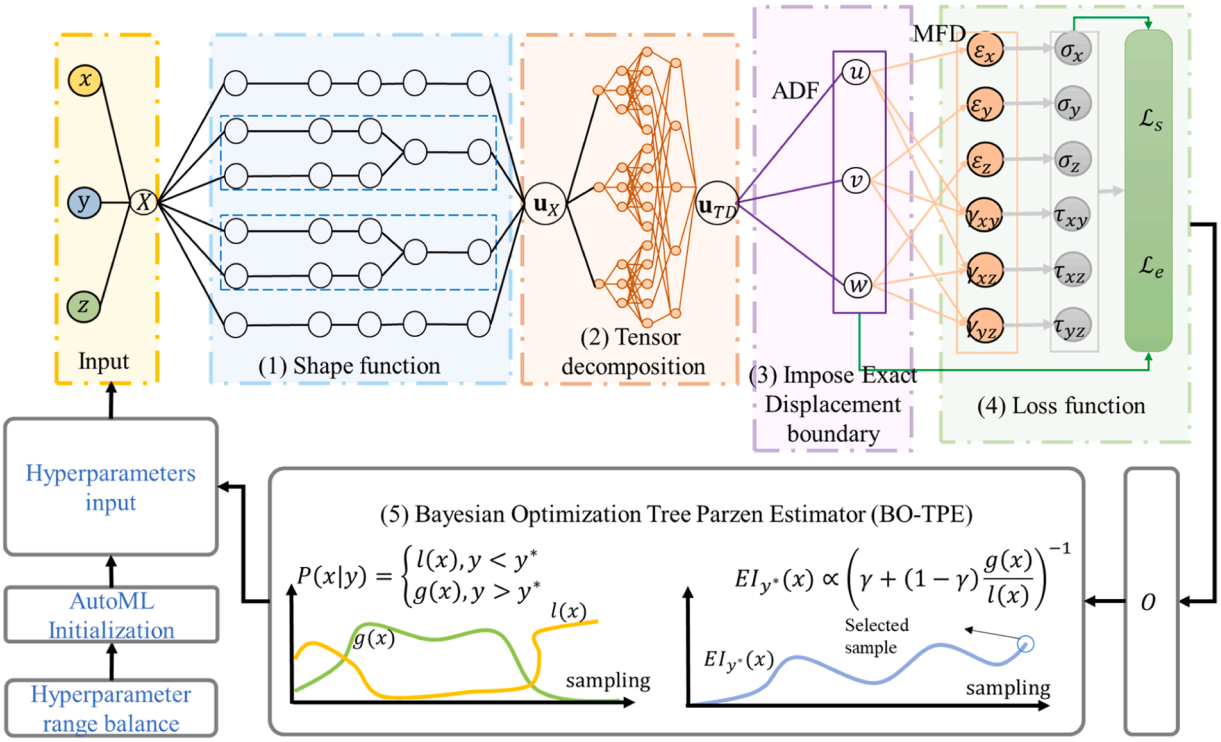



**Fig. 2.** Flowchart for addressing solid mechanical problems with the AEPINN framework. Note: MFD: Meshless Finite Difference, ADF: Approximate Distance Function (measured from any internal points to the boundary), $\mathscr{L}_e$: external work done, $\mathscr{L}_s$: strain energy.

irregularly shaped objects. Considering the hyperparameters of the model, there are the learning rate, mode of Tensor Decomposition (TD), mesh number of three directions, and interior sampling points in which the learning rate and rate decay steps mainly affect the model updating step per epoch. TD Mode, the number of shape function nodal points, and the number of interior sampling points influence the displacement prediction ability of EPINN block interactively. Besides, Meshless finite difference (MFD), as the differential method for calculating gradients, can obtain the gradient information (i.e. obtain strain field) from displacement field efficiently so that the whole framework can handle the mesh-free data sampled from 3D files.

Combined with the automated module, AEPINN can utilize the optimized structure to reduce the training time significantly and perform efficiently in solving the solid mechanics solution fields of irregular three-dimensional objects. In the following sections, we will introduce the EPINN block in our framework and Bayesian-Optimization Tree-Structured Parzen Estimator (BO-TPE) framework.

### 2.2. Exact Dirichlet boundary PINN (EPINN) framework

EPINN model consists of shape functions to formulate piecewise linear 1D solution field, tensor decomposition to reduce the complexity of data as well as construct 2D and 3D solution field, and Approximate Distance function (ADF) to enforce exact Dirichlet boundary condition to the domain. EPINN adopts the principle of least work to serve as the loss function, which is the strain energy minus the external work done. EPINN model utilizes four layers of convolutional neural networks (CNNs) to reformulate the shape function of 1D truss element based on previous research [16–18,20]. As shown in Eq. (1), the piecewise linear function of 1D linear truss element at node $I$ can be represented as a function of nodal coordinates $x$, where $x_I$ is the coordinate of the node $I$ and $N_I(x)$ is the shape function at this node.

$$N_I(x)=\begin{cases}\dfrac{x-x_{I-1}}{x_I-x_{I-1}}, & x_{I-1}\le x<x_I, \quad I\ge 1\\ \dfrac{x_{I+1}-x}{x_{I+1}-x_I}, & x_I\le x<x_{I+1}, \quad I\le n-1\\ 0, \text{otherwise}\end{cases} \tag{1}$$

For 1D truss cases, its global 1D shape function can be composed of numerous 1D elements based on nodal points. The number of elements $n$ in the 1D shape function, as one of the hyperparameters, should be defined before training the EPINN model. With increasing mesh number $n$, the computational time and accuracy can also be improved. Fig. 3 states that the neural network initializes weights and bias fixed to produce the 1D-shape function and capture the displacement at the separated nodal points. To establish this shape function globally, EPINN first will uniformly set the coordinates as $n$ nodes. An arbitrary point with the coordinate $x$ will be input to the adjacent node parts to estimate the shape function value of this coordinate $x$. By utilizing the Rectified Linear Unit (ReLU) activation function, the 1D shape function can be derived as shown in Eqs. (2)-(3), where $\mathscr{A}$ is the ReLU activation function.[32]. As shown in Fig. 3, constructing a four-layer neural network allows the shape function of a 1D truss element to be reformulated exactly, and the trainable weights represent the nodal displacement respectively. The global 1D-shape function can be obtained as shown in Eqs. (4)-(6).

As mentioned before, $\mathscr{A}$ is the activation function Relu to create the shape function [16,21]. After obtaining the global shape function for the sampling point, we can derive the displacement field expression considering nodal predicted displacements as Eqs. (4) and (5). The global displacement function can predict the displacement of this point by multiplying the global shape function with the predicted nodal displacement at position $I$ next to the sampling point.

$$N(x)=\sum_{I=1,2,\ldots,n} N_I(x) \tag{2}$$

$$N_I(x)=\begin{cases}\mathscr{A}\left(\dfrac{-1}{x_{I+I}-x_I}\mathscr{A}(x-x_I)+1\right), I=1\\ \mathscr{A}\left(\dfrac{-1}{x_I-x_{I-I}}\mathscr{A}(-x+x_I)+1\right)+\mathscr{A}\left(\dfrac{-1}{x_{I+1}-x_I}\mathscr{A}(x-x_I)+1\right)-1, \quad 2\le I\le n-1\\ \mathscr{A}\left(\dfrac{-1}{x_I-x_{I-I}}\mathscr{A}(-x+x_I)+1\right), I=n\end{cases} \tag{3}$$

$$u(x)=\sum_{I=1,2,\ldots,n} u_I(x)N_I(x) \tag{4}$$

$$u_I(x)=\begin{cases}ampamp\mathscr{A}\left(\dfrac{-1}{x_{I+I}-x_I}\mathscr{A}(x-x_I)+1\right)u_I, I=1\\ \&\left(\mathscr{A}\left(\dfrac{-1}{x_I-x_{I-I}}\mathscr{A}(-x+x_I)+1\right)-0.5\right)u_I+\left(\mathscr{A}\left(\dfrac{-1}{x_{I+1}-x_I}\mathscr{A}(x-x_I)+1\right)-0.5\right)u_I, 2\le I\le n-1\\ ampamp\mathscr{A}\left(\dfrac{-1}{x_I-x_{I-I}}\mathscr{A}(-x+x_I)+1\right)u_I, I=n\end{cases} \tag{5}$$

In EPINN, displacement prediction is mainly conducted by combining Tensor Decomposition (TD) with shape functions. TD is a method that can significantly reduce the complexity of the dataset from higher dimensions to lower dimensions, which is considerably beneficial

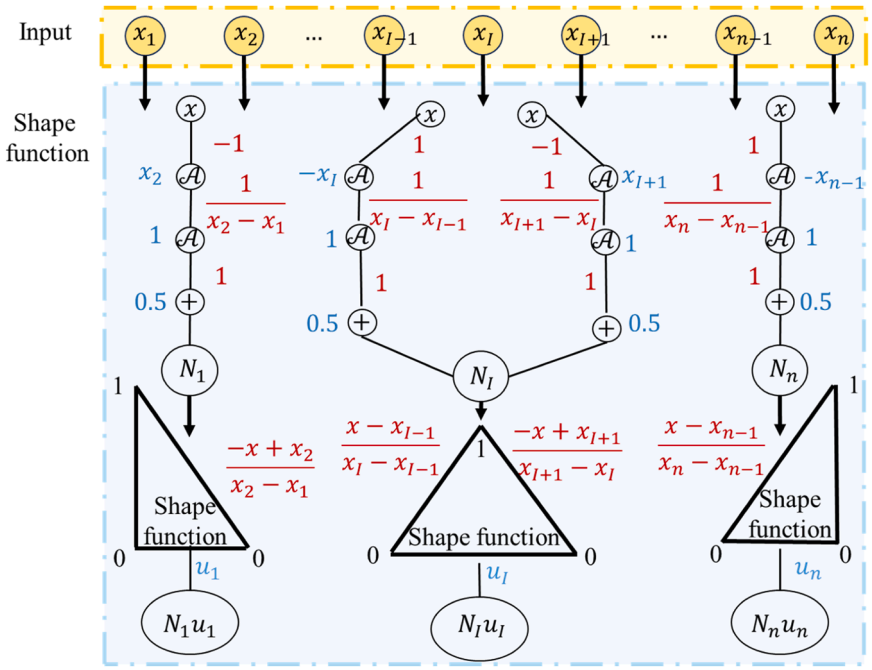



**Fig. 3.** The 1D-Shape function of the EPINN module. Note: $x_I$: $i$th nodal coordinates, $N_I$: shape function, $u_I$: predicted displacement; the green and purple color numbers are the bias and fixed weights respectively.

in addressing the 3D static mechanical analysis problems by reducing the computational costs[17,18,20]. Fig. 4 illustrates the TD process of EPINN, where 1D shape functions are utilized to calculate weighted predicted values so that the three-dimensional displacement can be predicted by the trainable weights in the TD layer. As shown in Eq. (6) and Fig. 4, effective as it is, TD also creates mode $Q$ as a new hyper-parameter, which needs to be considered. More details can be discovered in the EPINN paper [20].

$$\begin{aligned}\mathbf{u}_{\mathrm{TD}}(\boldsymbol{x}) &= \mathbf{u}_{\mathrm{TD}}(x,y,z)\\ &= \sum_{q=1}^{Q}\left(\sum_{I=1}^{n_1} N_I(x)\beta_I^{(q)}\right)\left(\sum_{J=1}^{n_2} N_J(y)\gamma_J^{(q)}\right)\left(\sum_{K=1}^{n_3} N_K(z)\theta_K^{(q)}\right)\end{aligned} \tag{6}$$

### 2.3. Bayesian-optimization tree structured parzen estimator

AutoML is the technique that automatically helps researchers simplify the machine learning workflow, including data preparation,

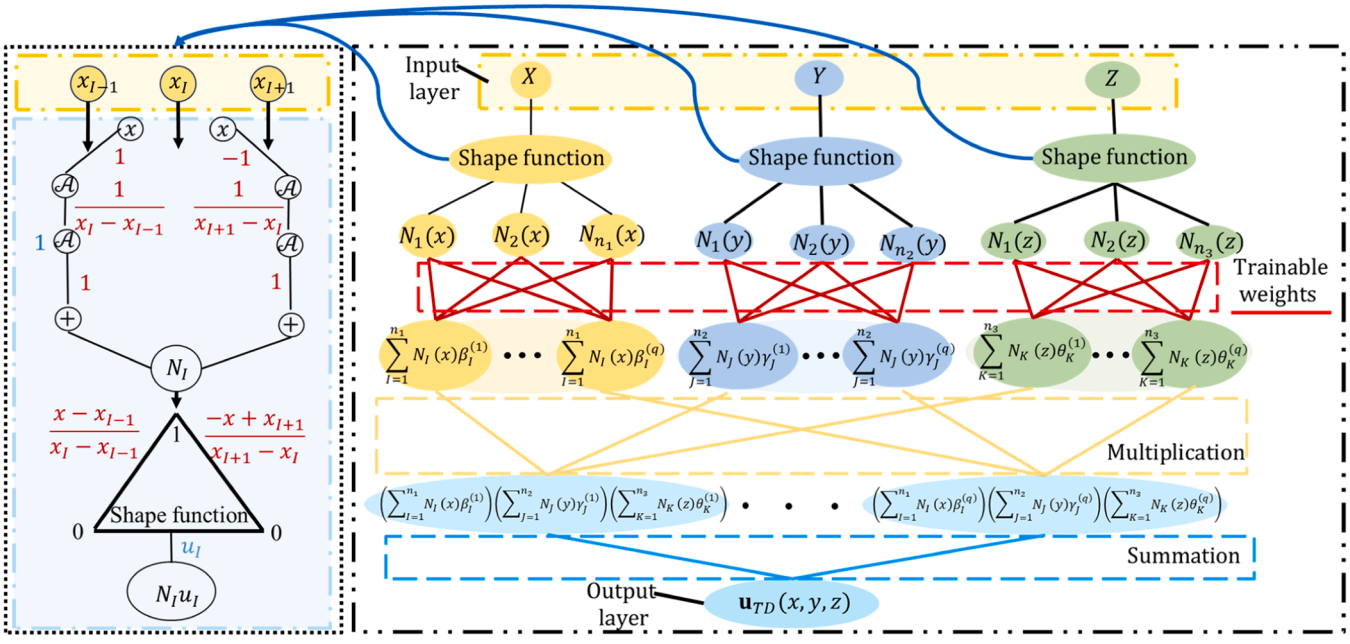



**Fig. 4.** Tensor Decomposition formulated by neural network. Note: $N_I(x)$, $N_I(y)$, $N_I(z)$: shape function of $x$, $y$, $z$; $\mathbf{u}_{\mathrm{TD}}(x,y,z)$: solution displacement field of the coordinates; $\beta_I^{(q)}$, $\gamma_J^{(q)}$, $\theta_K^{(q)}$: the $q$th trainable weights of the tensor decomposition layer.

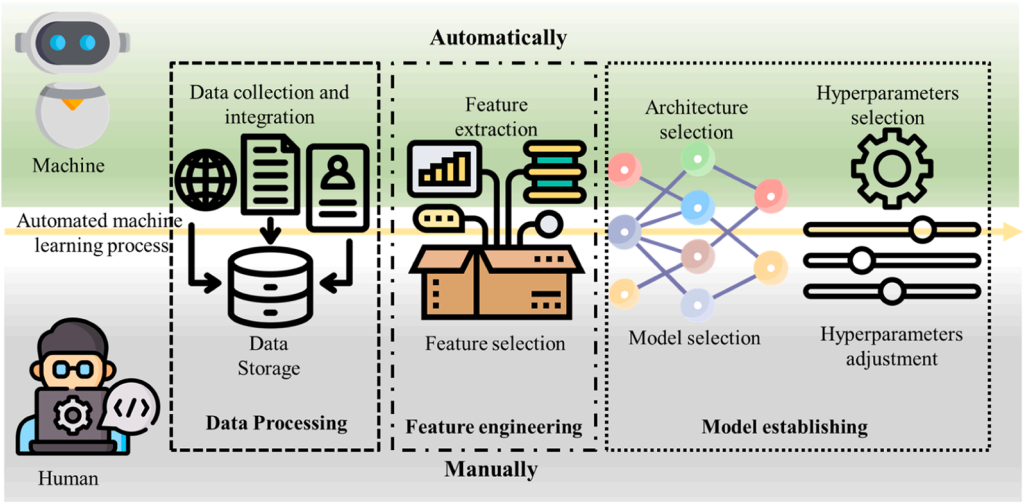



**Fig. 5.** The framework of Automated Machine Learning with Human Machine interaction.

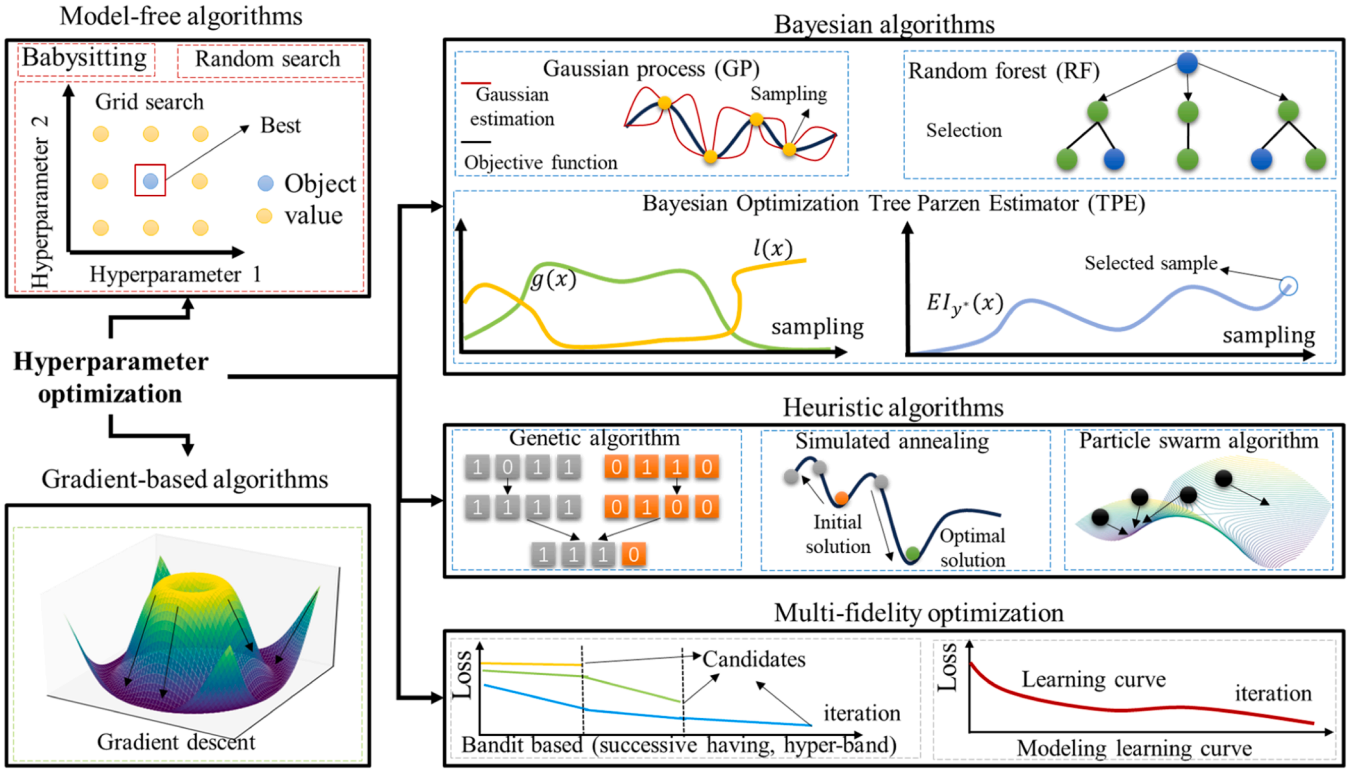



**Fig. 6.** Types of hyperparameter optimization methods. Note: $EI_{y^*}(x)$ is the expected improvement; $l(x), g(x)$ are density functions of the bad and good performances.

feature engineering, and model establishing [33]. Some software designed for AutoML, including Auto-Weka, Auto-sklearn, and Auto-ml, are developed in Python for the simplification of machine-learning tasks [34]. Fig. 5 describes the three stages of AutoML, including data processing, feature engineering, and model establishment. Since the pipeline of AutoML is mainly classified into three stages as displayed in Fig. 5, various algorithms are proposed for each specific aspect of the research [35]. Conventional PINN architectures in addressing complicated solid mechanics may be time-consuming and very difficult to converge after a long time of training. Therefore, AutoML integrated PINN framework is rarely reported. However, with the remarkable development of PINN architectures, establishing AutoML to explore the potential of PINN as solvers is becoming critical for achieving effective speedup in preparing to solve real-scale problems. In bridging the gaps, this study develops the AEPINN model to estimate the performance of the PINN framework in solving solid mechanics problems accordingly.

PINN solvers do not need to cover large datasets and features mainly be defined based on physical laws. Especially, for solid mechanics, the first two steps are not crucial in improving the performance of PINN solvers. Hence, considering the hyperparameter optimization things, Fig. 6 illustrates techniques to optimize the model hyperparameters during model establishing process, including Model-free algorithms, gradient-based algorithms, heuristic algorithms, and multi-fidelity optimization [36]. Model-free algorithms, including babysitting, random search, and grid search, can be established accordingly by priori knowledge without the specific model, which is widely used with

limited performance. Gradient-based algorithms mainly apply a gradient descent algorithm during the hyperparameter updating process, where gradients of error with respect to hyperparameters are hard to obtain from PINN solvers and will create new parameters. Despite the high efficiency of solving some specific tasks, heuristic algorithms are highly related to the specific initial solution quality and the applicability and adaptivity of the heuristic section to the studied problems. Successive halving and hyperband in the multi-fidelity optimization part are efficient for cases with a stable converging training process but not suitable for PINN solvers due to the unapparent initial convergence conditions in the learning curves of differential solvers. Compared with the aforementioned algorithms, Bayesian optimization (BO) is another important hyperparameter optimization algorithm that is efficient for most problems. Instead of random searching, BO is a history-informed hyperparameter optimization algorithm. BO can explore and exploit the unsampled areas and promising regions to improve the surrogate function between the hyperparameters and losses, minimizing the chances of skipping the optimal part based on historical experiences [37], which may be more appropriate for PINN solvers.

BO can be formulated by applying sequentially surrogate model updating to dynamically utilize the historical information to decide the position of the next sampling point so that the errors between the surrogate model and the true function between hyperparameters and loss can be minimized. Sequential model-based optimization (SMBO) of BO mainly consists of three types of algorithms, including Gaussian Process (GP), Random Forest (RF), and Tree-structured Parzen estimator (TPE), and SMBO is designed to solve the black box function from hyperparameters to the loss function [30]. RF is more efficient for simple optimization problems by using regression and classification. GP predefines the Gaussian relationships between the black box function and the hyperparameters, while TPE does not need to define the relationship of variables [38,39]. Thus, TPE is more suitable for the EPINN block.

Composed of several components, including the domain, surrogate model, acquisition function, true loss function, and historical information, SMBO is the model-based method used to optimize the surrogate model to fit the true loss function that can reflect the relationships between hyperparameters and the losses of the trained deep learning structure. Initializing the surrogate model $S$ as $p(y|x)$ and acquisition function at the beginning. In processes, set $y^*$ is the best value of current observed results $y^* = \min\{f(x_i), x_i \in X\}$, where $I$ is the set of sampling examples. GP will calculate the expected improvement as the acquisition function: $EI_{y^*}(x) = \int_{-\infty}^{+\infty} \max(y^* - y, 0)p(y|x)dy$, where the $EI_{y^*}(x)$ means the expectation results of the assumed surrogate function $p(y|x)$ that is modeled by applying the GP model. GP will select the next exploration or exploitation action based on the estimated expected improvement $EI$. The region with a better mean value or higher uncertainty will be selected respectively. Then, SMBO will test this trial to obtain the value of the true loss function to update the historical information for the upcoming next step.

Similar to the GP, TPE will apply the Bayesian rule by calculating the probability of $x$ with the observed condition $y$, $p(x|y)$ instead of the $p(y|x)$, given $x$ is the hyperparameter and $y$ is the value of the true loss function. The relationship of $p(x|y)$ and $p(y|x)$ is the Bayesian rule as $p(y|x) = \frac{p(xy)}{p(x)} = \frac{p(x|y)p(y)}{p(x)}$. As shown in Fig. 6, TPE defines the $p(x|y)$ as the piecewise function with the two density functions $l(x), g(x)$ to first classify the remained samples into bad and good performance sets as Eq. (7) displayed, where $y^*$ is the threshold that will be selected by using the quantile $\gamma$ so that $p(y < y^*) = \gamma$. Thus, the expected improvement $EI$ of the TPE method can be obtained as Eq. (8). The next candidate will be selected based on the value of $\frac{(1-\gamma)g(x)}{l(x)}$. TPE will choose the sample with the highest expected improvement $EI$ during each iteration. Hydra is a framework to control complicated configured machine learning problems with only configuration files to control most parameters which is easy to use for multiple experiments which is what exactly automated machine learning needed. Optuna is an optimization tool that can be integrated with hydra to automatically control the whole AutoML procedure without any extra required command for users to implement. Besides, Optuna integrated various Bayesian Optimization algorithms which are easier for researchers to develop their own frameworks. In this section, we choose Hydra-Optuna as the base platform to develop the AEPINN framework.

$$p(x|y) = \begin{cases} l(x), if y < y^* \\ g(x), if y \geq y^* \end{cases} \tag{7}$$

$$\begin{aligned} EI_{y^*}(x) &= \int_{-\infty}^{y^*} (y^* - y)p(y|x)dy \\ &= \frac{\gamma y^* l(x) - l(x)\int_{-\infty}^{y^*} p(y)dy}{\gamma l(x) + (1-\gamma)g(x)} \propto \left(\frac{1}{\gamma + \frac{(1-\gamma)g(x)}{l(x)}}\right) \end{aligned} \tag{8}$$

### 2.4. Automated machine learning EPINN (AEPINN) framework

As mentioned earlier, this framework utilizes Eq. (9), the principle of least work applied in [18,32,40] as the loss function, where $\mathbf{f}$ denotes the body force, $\mathbf{u}$ denotes the predicted displacement field, and $\mathbf{t}$ represents the traction applied to the boundary. In Eq.(10), the displacement field is predicted by the trained EPINN framework $\mathbf{u}_{TD}$ and then forced to apply the Dirichlet boundary of the region [20,21]. Through this method, when minimizing the energy loss, the boundary $\mathbf{u}^*_{\partial\Omega_D}$ is satisfied automatically by applying this ADF module $\phi_{ADF}$. As Eq. (11) depicted, the objective is to discover the appropriate weights $\widehat{W}$ of the neural networks to minimize the loss $\mathscr{L}_{energy}(\mathbf{u}; \mathbf{f}, \bar{\mathbf{t}})$ during training. We don't adopt data loss since in reality it is usually difficult to obtain full fields of the studied structures. Therefore, many engineering problems cannot provide abundant data for researchers to establish a dataset for machine learning algorithms to learn. Besides, the transfer learning ability of physics-driven methods can be further enhanced if the model is efficient in solving mechanics problems without labeled data, indicating the possibility of learned physical laws behind the performance.

$$\mathscr{L}_{energy}(\mathbf{u}; \mathbf{f}, \bar{\mathbf{t}}) = \frac{1}{2}\int_{\Omega} \sigma(\mathbf{u}) : \varepsilon(\mathbf{u})d\Omega - \left(\int_{\Omega} \mathbf{u} \bullet \mathbf{f} d\Omega + \int_{\partial\Omega_N} \mathbf{u} \bullet \bar{\mathbf{t}} d\partial\Omega_N\right) \tag{9}$$

$$\mathbf{u}(u, v, w) = \mathbf{u}^*_{\partial\Omega_D} + \mathbf{u}_{TD} \bullet \phi_{ADF} \tag{10}$$

$$\widehat{W} = \underset{w}{\operatorname{argmin}}\left(\mathscr{L}_{energy}(\mathbf{u}; \mathbf{f}, \bar{\mathbf{t}})\right) \tag{11}$$

In AEPINN structure, the energy functional loss will be the machine learning loss to automatically update the trainable weights within the EPINN module. Fig. 7 illustrates the gradient backpropagation process within the AEPINN framework. The only trainable weights are within the Tensor decomposition part which corresponds to the calculation from shape function to the tensor decomposition predicted displacement fields $\mathbf{u}_{TD}$. $\mathbf{u}_{TD}$ will be further forwarded to obtain the final displacement fields as the predicted solution fields $\mathbf{u}(u, v, w)$. With the displacement fields, the strain and stress results can be obtained through applying the meshless finite derivatives. Through the final objective function, the Bayesian optimization section will evaluate the performance of the hyperparameters and adjust the hyperparameter selection results accordingly. Since the Bayesian process is gradient free method and neural network layers that form shape functions are fixed with untrainable weights and bias, EPINN module will be easily trained with only the TD part in Eq. (6).

In the BO-TPE process, the objective is to minimize the errors with respective hyperparameters. As shown in Eq. (12), for the EPINN module, the objective function $O$ will be optimized by applying the BO-TPE process to discover the mesh number $n$, decomposition mode $q$, learning rate $l_r$, the number of sampling points $n_{sample}$, and the learning rate decay

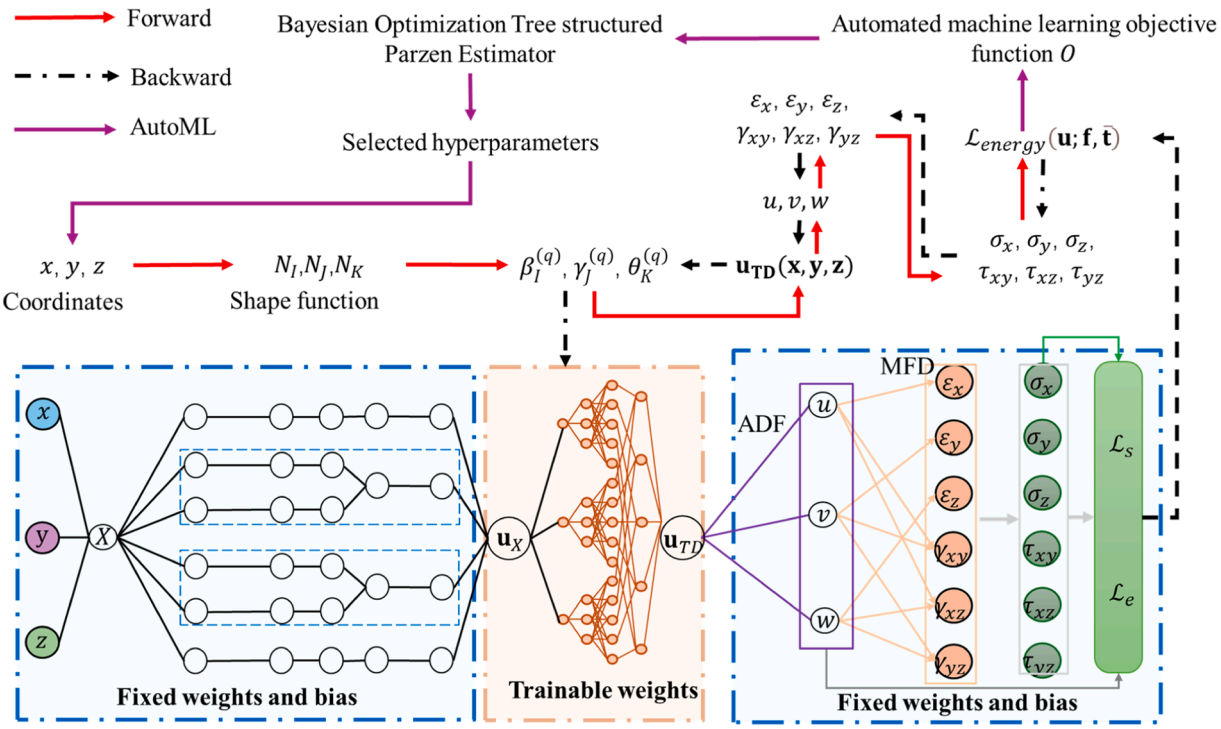



**Fig. 7.** Backpropagation process of EPINN module during AutoML procedure.

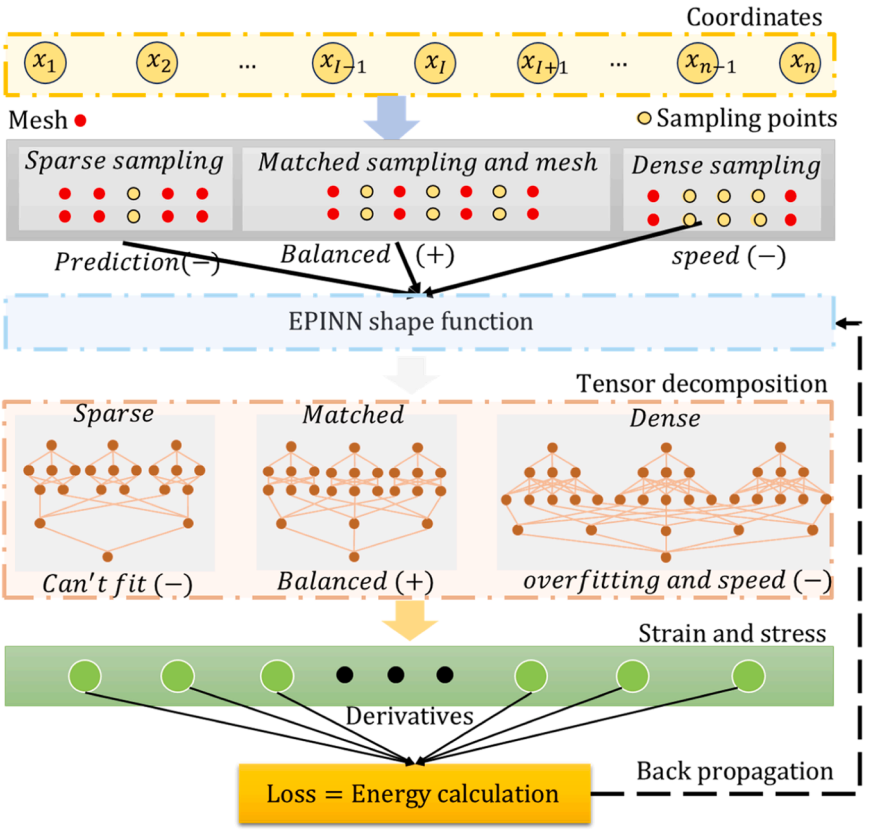



**Fig. 8.** Hyperparameters matching influences of the AEPINN framework.

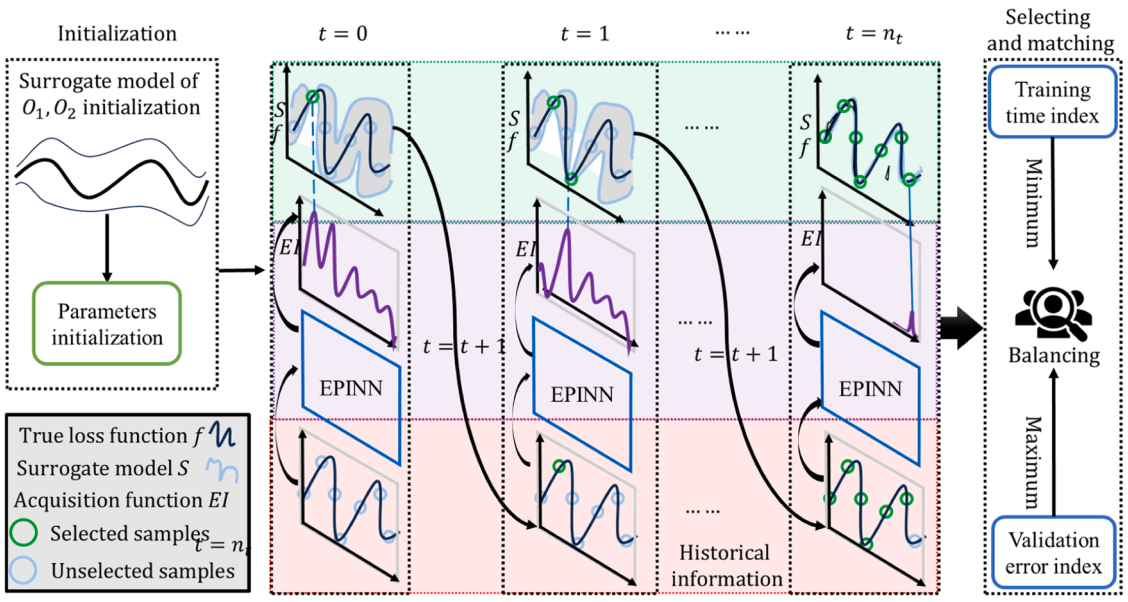



**Fig. 9.** AEPINN gradient-free pipeline.

steps $s_{decay}$ that can reduce errors. $O$ denotes the minimum average L2 relative error. $\mathbf{u}_{ipred}$ is the predicted displacement at $i$th direction and $\mathbf{u}_{itrue}$ is the corresponding reference value.

$$O = \underset{n,\ q,\ l_r,\ n_{sample},s_{decay}}{\operatorname{argmin}} \frac{1}{n'} \sum_{i=1,2,n'} \sqrt{\frac{\sum \left(\mathbf{u}_{ipred} - \mathbf{u}_{itrue}\right)^2}{\sum \mathbf{u}_{itrue}{}^2}} \tag{12}$$

To consider the AEPINN framework, it is of great necessity to ensure the fitting process during training with appropriate hyperparameters. In FEM, the mesh number $n$ can positively influence the accuracy of FEM results. However, sampling, just like the adaptive finite element, is important in addressing high-dimensional problems [41]. Since AEPINN is a mesh-free method, enough sampling points from the objects are necessary, especially for irregularly shaped objects. To improve this, this research applies the Signed distance function (SDF) to so that AEPINN can sample complicated objects by using STL files obtained from various software or scanned data.

Thus, the mesh number $n$, tensor decomposition mode $q$, and the sampling points $n_{sample}$ are related to each other. Normally, the common method to improve accuracy is to simultaneously improve the mesh number and tensor decomposition mode. However, the unbalanced size of sampling points and mesh number will influence the efficiency and accuracy. With high values of the mesh number and the mode, the model holds a better ability to predict the displacement field, but there may also be overfitting issues during the training sampling points. If the mesh number and the mode values are relatively small, this framework will not be able to capture the features of points fully with an exceeding amount of sampling points.

Considering this, we propose the sampling points and relationship framework to accelerate the automated machine learning training speed along the way. As Fig. 8 depicts, the sparse sampling and tensor decomposition can impose negative effects on the EPINN framework. Although a higher number of sampling points $n$ and decomposition mode $q$ can improve the predicting ability of EPINN, it will also increase the training time extremely and induce overfitting problems. Compared with the aforementioned hyperparameters, the matched hyperparameters mesh number $n$, the decomposition mode $q$, and the number of sampling points $n_{sample}$ will significantly contribute to the convergence speed and accuracy of EPINN module.

The automated machine learning framework applied in this EPINN module is composed of several key steps, as shown in Fig. 9. At the beginning stage, the surrogate model of the objective function $O$ is initialized before the BO-TPE framework. Similar to the SMBO process, in this framework, EPINN module will be the bridge between the hyperparameters inputting and expected improved factors outputting by calculating the energy loss function, respectively. After $n_t$ trials, the energy loss function can be obtained accordingly. Normally, the obtained hyperparameters cannot balance the training time and validation errors well. To fix this, the previously proposed hyperparameter matching balance method can be applied to match the mesh number, resolutions, and interior sampling ranges subsequently so that the selected hyperparameters can maintain the prediction accuracy as well as efficiency simultaneously.

## 3. Performance of AEPINN for solving solid mechanics

This section solves four solid mechanics problems by applying the AEPINN framework without labeled data, including a 2D plane stress problem, a 3D bracket traction case, a 3D complex frame structure and a hyperelastic rubber cube solid mechanics problem. These cases are conducted and tested based on the Nvidia Modulus platform [42], an open machine-learning environment that is installed on the supercomputer SQUID system from Osaka University and one local Personal Computer (PC) with GPU RTX 3080Ti. All cases of AEPINN are retested on a local PC with the GPU RTX 3080Ti 11 GB. The referred PINN architecture is set with a neural network of 6-layer depth, 512 neurons width, and the SILU activation function to avoid the gradient dead. For the AEPINN model, as mentioned before, hyperparameters including the mesh number $n$, decomposition mode $q$, learning rate $l_r$, the number of sampling points $n_{sample}$, and the learning rate decay steps $s_{decay}$ will be considered [43]. For fair comparisons, the ABAQUS implicit solver FEM results are obtained from the single CPU Intel i7–13700K using double precision with General Purpose GPU (GPGPU) acceleration using Nvidia RTX 3080Ti accelerator. The EPINN architecture results are simulated based on the provided hyperparameters [20], which may have some acceptable differences due to the random initialization torch seed. RTX 3080Ti, released on 3rd June 2021 which is comparably computational efficient as the general devices in 3D games and scientific computing with affordable price, was picked and installed with Pytorch framework rather than Nvidia Modulus for a fair comparison of the performance of proposed AEPINN on a normal GPU device. For all cases below, the performance diagrams in point cloud form are plotted based on the local

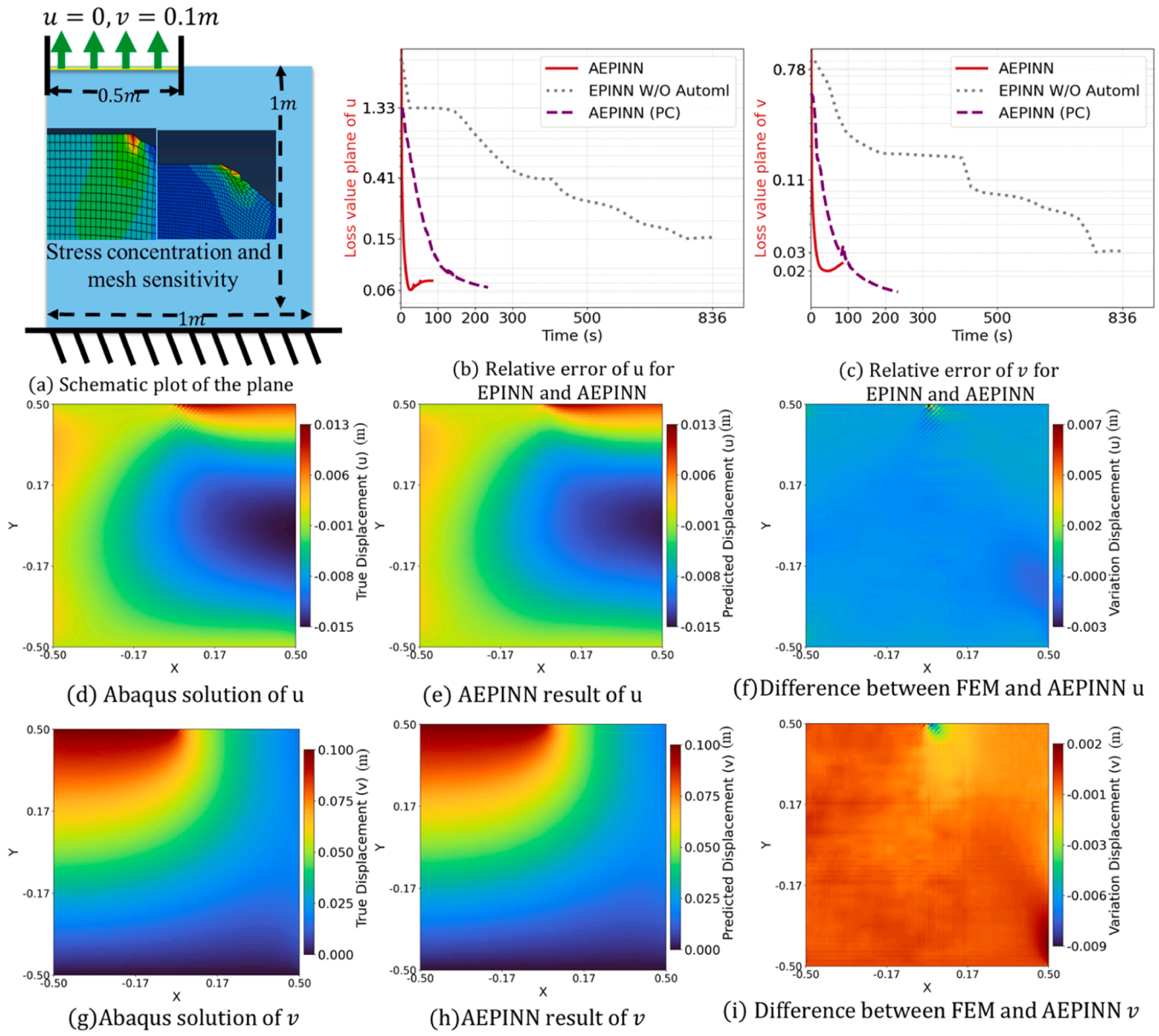



**Fig. 10.** The performance of AEPINN for simulating plane stress compared with Abaqus.

PC with one GPU 3080Ti results on FP32 precision to analyze the adaptivity of this framework. The training loss differences between FP32 and FP64 on SQUID supercomputer and Nvidia Modulus platform are also compared in the learning curve figures. Since TF32 is for improving the training speed with reduced precision, TF32 will not be used in this section for testing the model ability with enough precision.

In this section, in total four solid mechanics problems are studied including a complicated panel stress problem, a 3D bracket traction cases, one frame case and a hyperelastic problem which is a cube rubber with Young's Modulus 1.1572 MPa and Poisson ratio 0.499 for testing the proposed AEPINN method in solving nonlinear problem ability. Types on Abaqus for the panel stress, frame, and cube rubber cases are CPS4R, C3D8R, and C3D8RH respectively. For the cube rubber case, conventional FEM cannot deal with high Poisson ratio directly with the C3D8R element, which is replaced with C3D8RH, the hybrid element and allows large deformation to ensure Abaqus can solve this problem.

Non-dimensionalized approaches and scale methods are required for preprocessing data, especially for specific problems with the second and third cases. In the bracket case, the traction and stress are conducted through non-dimensionalized approaches as $\sigma_n = \frac{L_0}{Gu_0}$, where G is the 0.01 scaled from original shear modulus to avoid large values during training. The coordinates in the frame case are scaled to 1/13 to avoid the convergence problem during training.

The total AEPINN simulation time for automated machine learning for the 3D bracket and frame are around six hours and three hours. Because we are conducting hyperparameters optimization only for these two problems and the respective nonlinear cube rubber and plane problems can be directly tested with the optimized results. The AutoML is not needed to be rerun for every model. Therefore, considering the limited automated training time on several trials with multiple cases, it would be much better to compare the optimized results for the several trials counting for around thirty hours can be regarded as a pre-trained step. The optimized hyperparameters have been conducted on transferring from the frame case to the cube rubber case for performance verification, which also illustrates the importance of conducting AutoML as a pre-trained step for the continuous application of artificial intelligence in solving nonlinear solid mechanics problems.

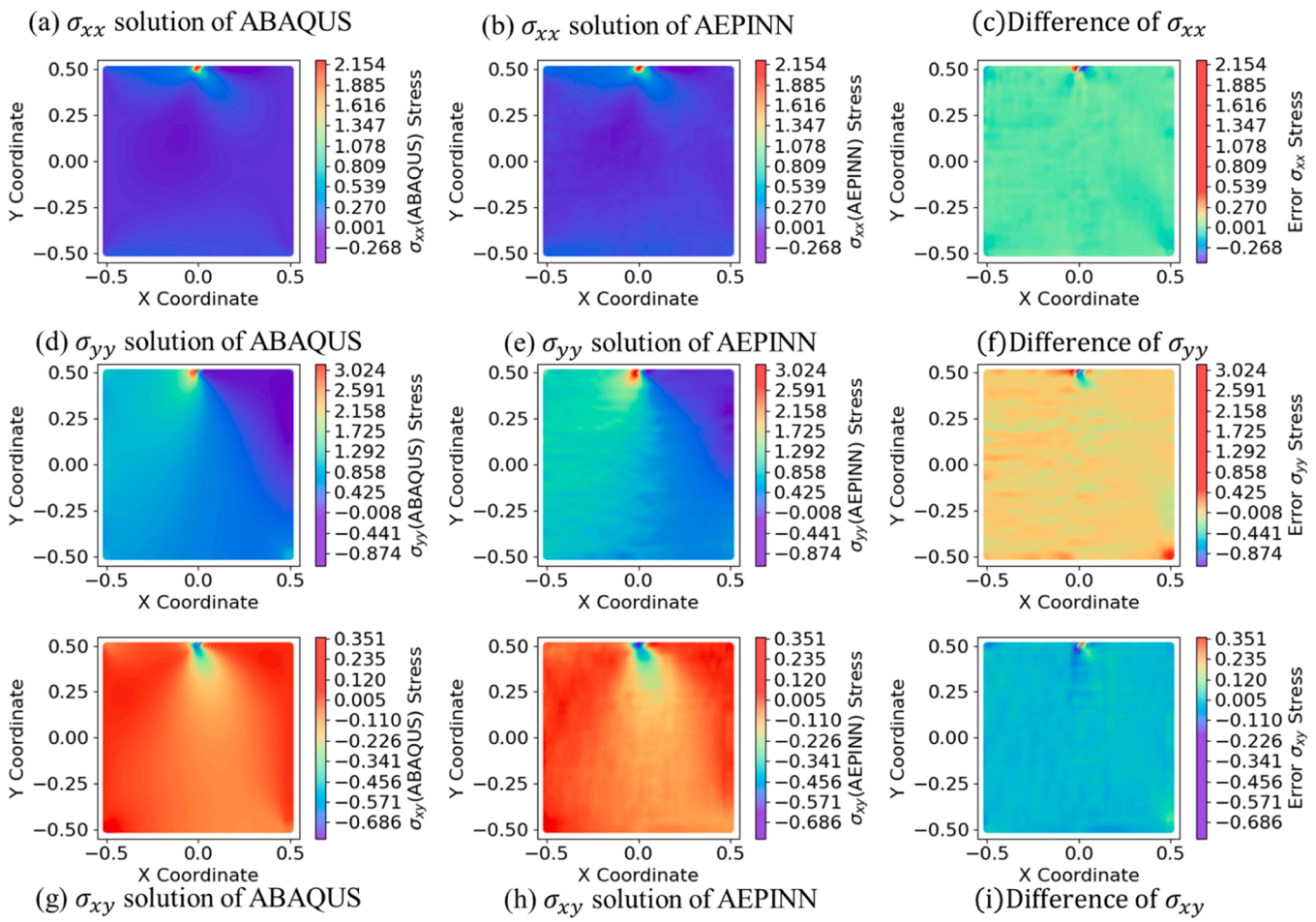



**Fig. 11.** The performance of AEPINN for simulating plane stress fields.

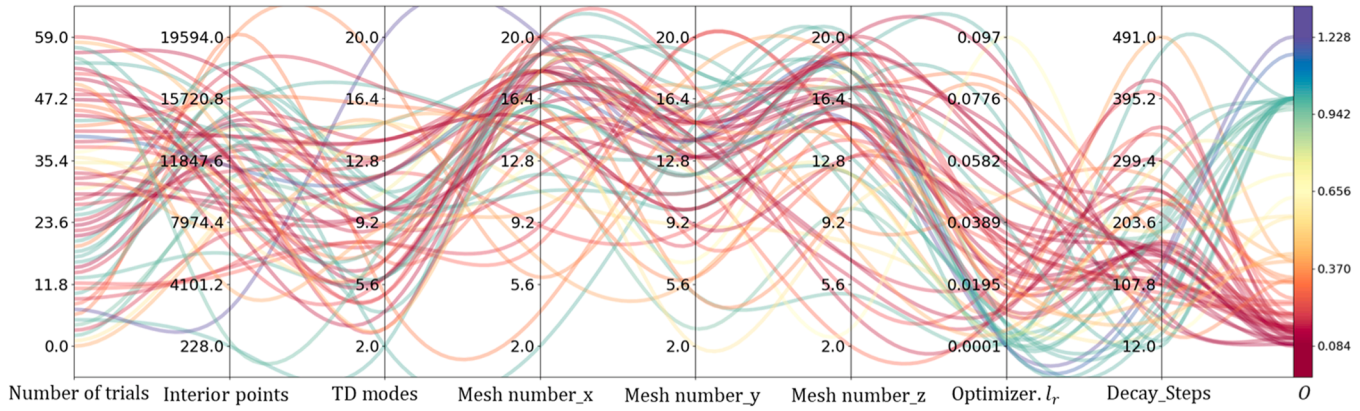



**Fig. 12.** Automated machine learning results of EPINN for the bracket.

### 3.1. Plane stress panel under eccentric tension

The plane-stress panel simulation is reported as a benchmark baseline finite element case to testify to the performance of the PINN structures in solving solid mechanical problems [20,44]. Holding an equal length of width and height of 1.0 m, this plate is fixed at the bottom part and free in the other three directions. As shown in Fig. 11 (a), the material is elastic with Young's modulus of 10 MPa and a Poisson ratio of 0.2. The left half of the top part is restricted with a vertical displacement of 0.1 m and a horizontal displacement of 0.0 m. Rest parts are free to be deformed. Using the commercial solver ABAQUS as the comparison results, the plate model in ABAQUS is defined with a mesh size of 1/400.

For the conventional PINN, we set the PINN with a fully connected network with 6 the depth of layers and 512 the width of neurons to ensure the prediction capability and set the plate sampled with 40000 interior points to ensure sufficient data for PINN to address this benchmark problem. Despite the tuned hyperparameters, PINN cannot converge to stable results after a long time of training. Hence, PINN will not be displayed here. Since this is a stress concentration problem, to compare stress results, the hyperparameters of the AEPINN for this plate case are sampled with 60000 interior points to fit with the mesh number

$n$ of 50 and the mode $q$ of 13 to improve the prediction accuracy efficiently applied with the automated matching machine learning framework. We set the learning rate with an initialized value of 0.05 with a decay rate of 0.95 for 200 steps to accelerate the exploration process. To avoid overfitting, weight decay is adopted with the value of 0.001. Fig. 11 (b)-(c) displays the wall time training L2 relative error of the displacement $u$ and $v$ predicted results of the AEPINN architecture. While the single intel i7–10870H can solve this panel stress problem after 265 s, Fig. 11 (b)-(c) shows that L2 relative error of total displacement field from PC can be lower than 0.1 within 37 s which is nearly the same as ABAQUS results of 37 s of i7–13700K with General purpose GPU (GPGPU) acceleration of Nvidia RTX3080Ti provided by ABAQUS solver, achieving a speedup of seven times compared with ABAQUS speed 265 s of i7–10870H, 22 times the EPINN method of 836 s with better accuracy, and more than 160 times of the training speed of the conventional PINN 5957 s [20]. Fig. 11(d)-(i) presents the differences between the predicted displacement fields and the simulated results by ABAQUS. With fewer points of 60000 points compared with 400 × 400 in Abaqus and mesh number 50 of the EPINN architecture, AEPINN results can be close to the displacement fields of this 2D plane stress problem. Fig. 12 states the predicted stress fields of AEPIN compared with Abaqus results. As a stress concentration problem, the displacement and stress values will vary as the mesh size of Abaqus changes. Therefore, different mesh sizes will cause huge differences in certain unique points. As shown below, in the stress prediction fields, AEPINN can capture the displacement and stress fields aligned with the Abaqus results with a mesh size of 1/400 while some small errors exist at those stress concentration points due to the mesh size sensitivity of this specific problem. With Abaqus results of mesh size 1/200 or 1/800, the stress values results will be quite different. Compared with the fine-meshed Abaqus results, AEPINN can be close to the current stress fields except for errors of certain points.

### 3.2. Three-dimensional bracket

The three-dimensional bracket case is also simulated in this part to testify to the performance of the AEPINN framework in addressing the irregularly shaped solid mechanics problems. The conventional PINN with a fully connected neural network cannot converge within 24 hours and is not shown in the results. Fig. 12 presents the automated multivariable parallel diagram of AEPINN. For this 3D solid mechanics problem, 7 hyperparameters are selected to be optimized, including the number of interior sampling points, tensor decomposition mode, mesh number in three directions, learning rate, and its decay steps. Based on the matching principle, the range of sampling points, TD modes, and mesh number are limited to (100, 20000), (2,50), (2, 20), which is helpful to improve the automated machine learning efficiency. After 60 trials, the AEPINN objective value $O$ can converge to 0.07. Compared with the original automated machine learning without matching principle, the convergence speed is improved ten times more for smaller searching regions. Fig. 12 shows that the mesh number and TD modes tend to perform better in smaller value ranges for fewer training sampling points. The learning rate is much more randomly performed with a range of decay steps between 120 and 400. For AEPINN, the learning rate, decay steps, TD modes, and interior sampling points are set to 0.0085, 166, 32, and 17432 with a learning rate decay rate of 0.9. The mesh numbers of the three directions are set to 14, 20, and 14, respectively, to ensure the balance of efficiency and accuracy, which are well-

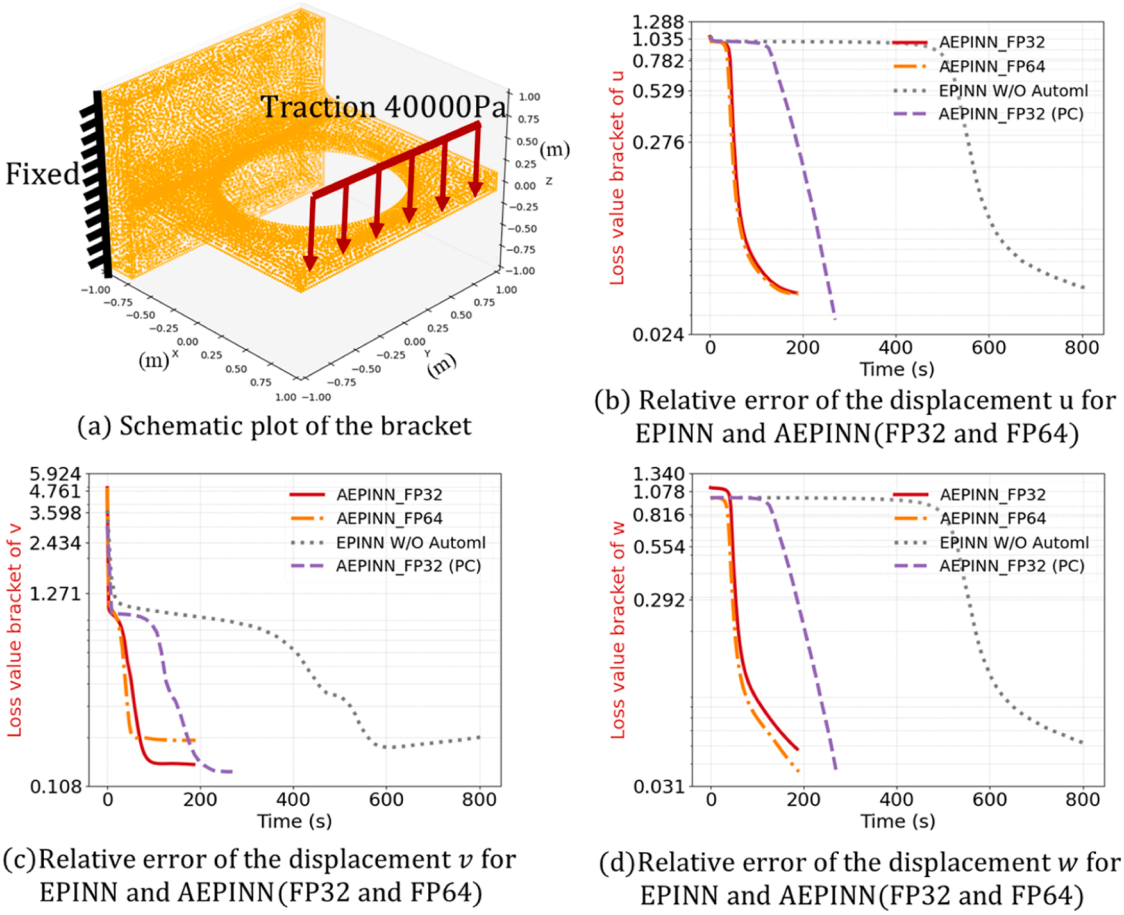



**Fig. 13.** Comparison of AEPINN performance with FP32 and FP64 precision and EPINN architecture with FP32 precision for the 3D bracket under uniform force boundary.

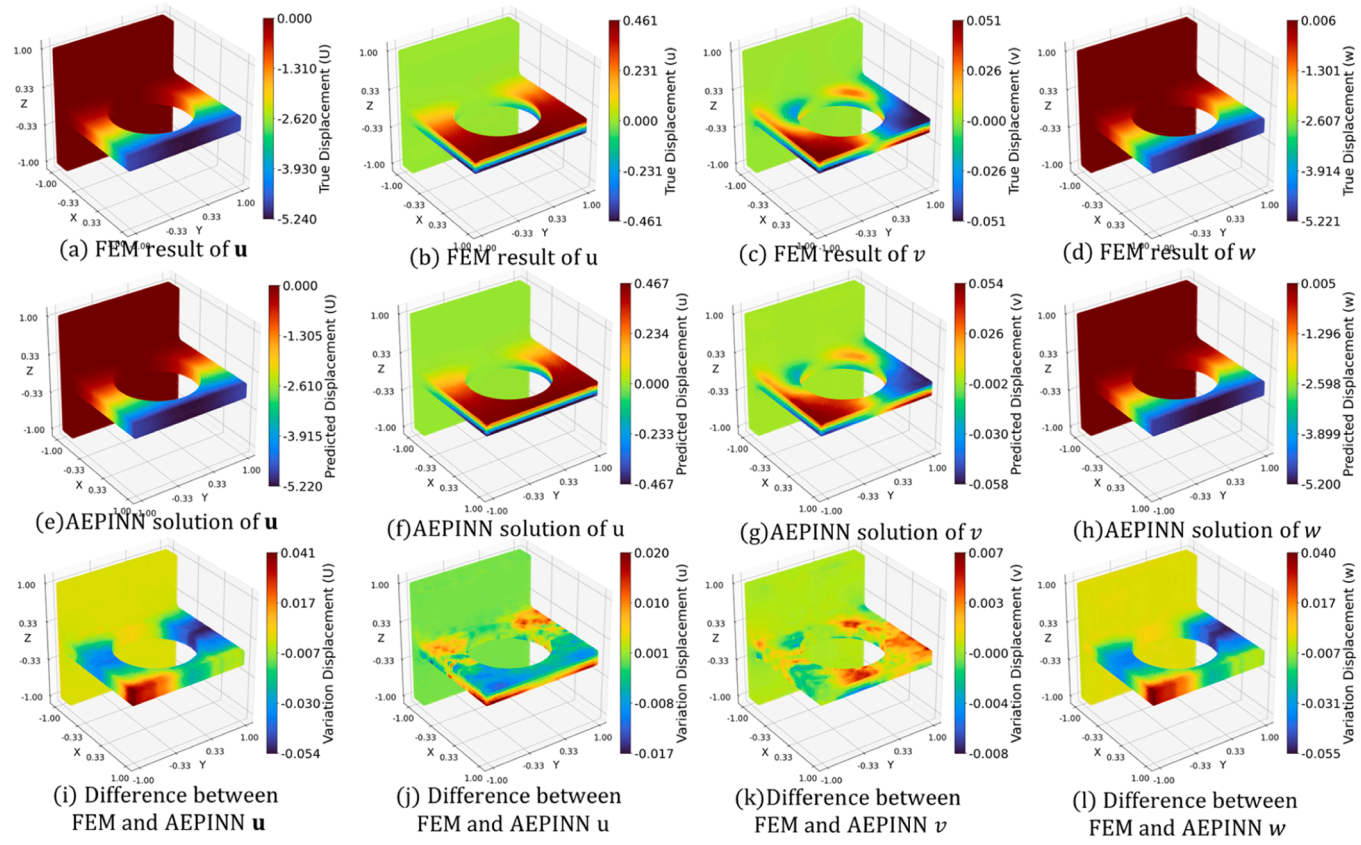



**Fig. 14.** Displacement solution field of AEPINN with FP32 precision for the 3D bracket (unit: mm).

tuned and selected based on the automated BO-TPE results in Fig. 12.

Fig. 13 (a) displays the concept figure of this bracket, which is fixed at the back face with the applied 0.4 MPa shear stress to the top surface edges. The rest parts are stress-free. This bracket has an equal height, width, and length range of 1.0 m. The Young's modulus of this case is 100 GPa with a Poisson ratio of 0.3. The reference results of this case are the FEM simulation with a mesh number of 0.003. Fig. 13 (b)-(d) exhibits the simulated results of AEPINN with different precision of FP32 and FP64. AEPINN with both precisions on A100 can converge within 200 s and achieves a speedup of four times of the three directions considering the EPINN architecture with 800 s in this simulation, five times speedup compared with the Matlab FEM speed 1140 s and more than 400 times of conventional PINN architecture speed 87052 s [20]. Compared with FP32, FP64 is a little bit faster but with similar FP32 results. The proposed AEPINN on PC also achieves a speedup se_of three times compared with EPINN and four times speedup compared with Matlab. Fig. 14 (a)-(d) displays the validated results of AEPINN in three directions and the total displacement field. Fig. 14 (e)-(h) presents the Matlab simulated results with a fine mesh size of 0.003 [20]. Fig. 14 (i)-(l) shows the variation of these four types of predicted and simulated results. It shows that AEPINN with optimized hyperparameters can well resolve this problem within minutes. It can be seen that the maximum overall displacement is about 5.24 mm, while the total displacement difference of predicted results from AEPINN only has some differences with a maximum value of about 0.054 mm, which is nearly a 1 % error. In the other three directions, the results are still within the acceptable range. Fig. 15 illustrates the stress fields of the 3D bracket case based on the displacement fields. Fig. 15 (a)-(c) denote stress in x direction and the main differences are at the area close to the upper and lower curve parts of the bracket. This may be due to the sampling points in these small regions are not enough. For Fig. 15 (c) the differences are comparably small in the top surface. Fig. 15 (d)-(f) are about the stress in y direction where similar errors exist due to the sampling problems. For other stress fields, stress values are too small to tell the differences. Thus, the third-row results are displayed with the Von-Mises stress field comparisons between the Abaqus and AEPINN results. The figures show that in the top and bottom surfaces the errors are located at the upper and lower curve parts similarly due to the sparse sampling points. The stress fields in the main directions are fitted close to the FEM result. However, there still remain some errors in the corners.

### 3.3. *Three-dimensional frame*

Despite the good performance of EPINN for the three-dimensional bracket, complex real-scale structures still require tremendous study to analyze the performance of artificial intelligence in understanding reality and predicting the correct displacement field for real cases. In this section, a three-story frame structure made of beams, columns, and slabs is established for the case study. The frame structure is applied with Dirichlet boundary conditions in the top surface along two directions with a displacement of 80 mm. This frame is a 3-storey case with slabs, beams, and planes together. It's 9000 mm high, 6500 mm long, and 6500 mm width. For this complex 3D simulation, the conventional PINN, EPINN architectures, and automated PINN cannot converge within minutes. Therefore, in this part, only AEPINN based on the framework will be testified to analyze the performance in addressing the static solid mechanical problems. Fig. 16 is the automated machine learning parallel diagram that depicts the variable relationships for the final objective $O_1$, the average relative L2 error. Considering the training efficiency, the mesh size values in three directions are set the same. Based on this BO-TPE process, the acceptable hyperparameters interior sampling points, TD modes, mesh size in three directions, learning rate, and decay steps are selected as 30000, 50, 13, 0.02, and 500. The reason why we select a sparse interior sampling point number is based on the performance in previous sections and Fig. 16, where AEPINN can converge to 0.2 with 25255 sampled points.

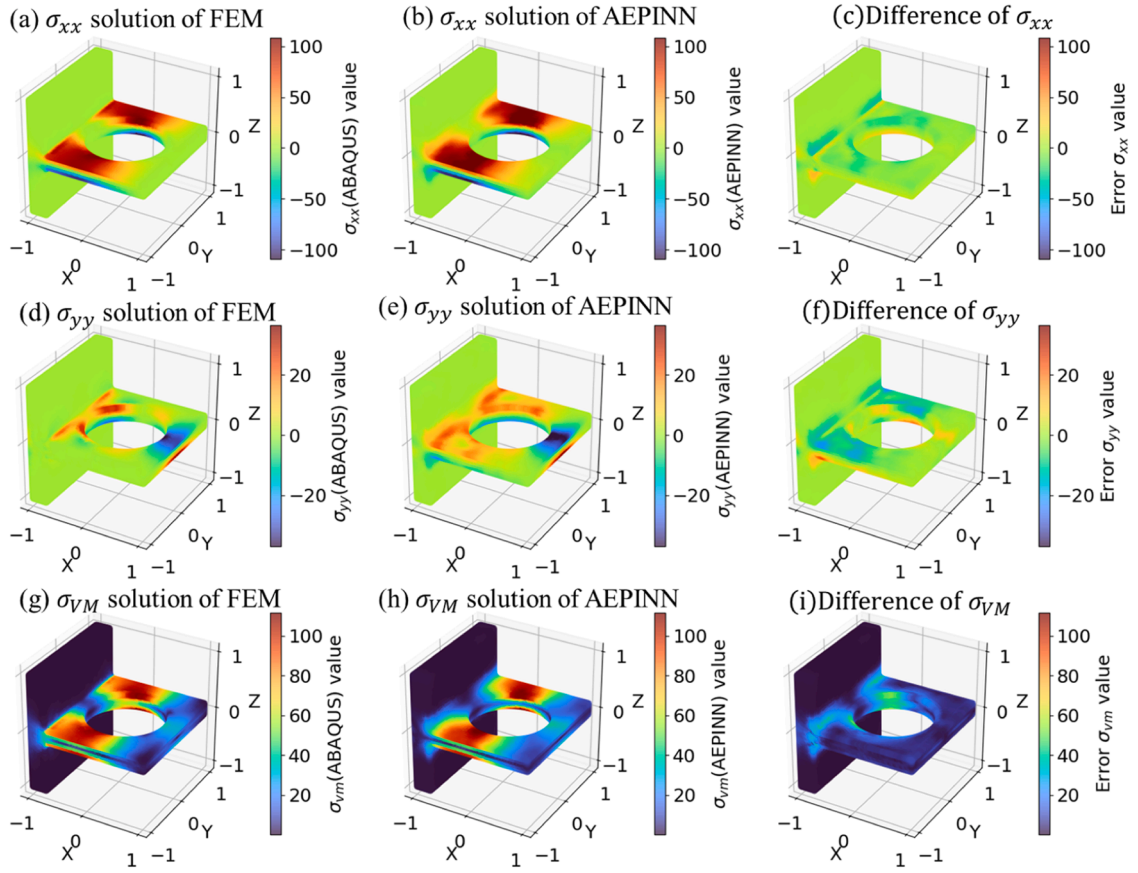



**Fig. 15.** Stress solution field of AEPINN with FP32 precision for the 3D bracket (unit: mm).

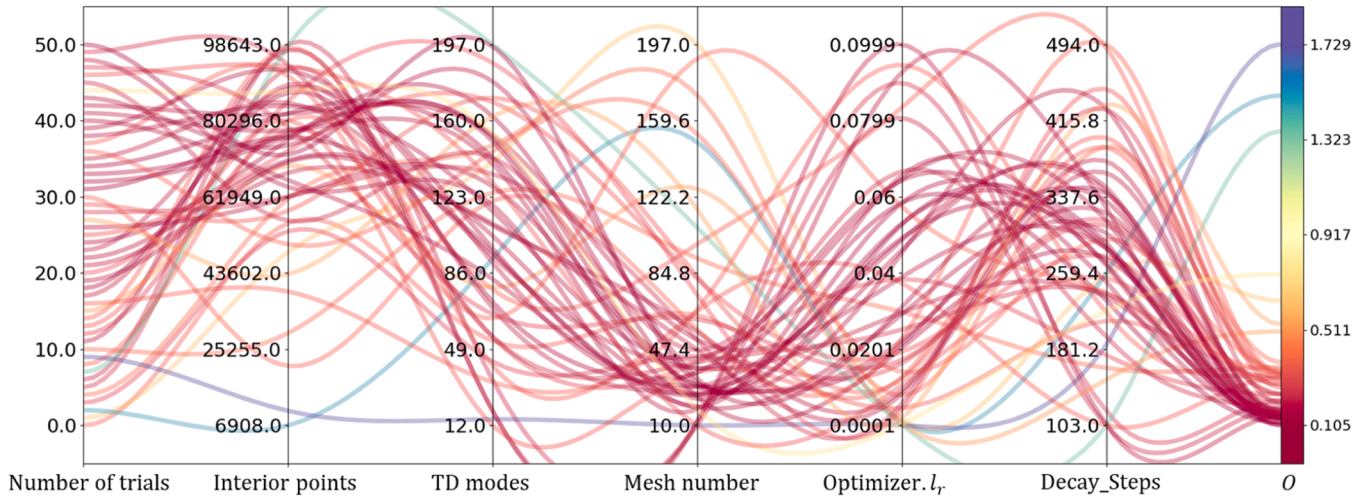



**Fig. 16.** Automated machine learning results of EPINN for the 3-storey frame.

Fig. 17 (a) displays the schematic plot of the 3-story frame structure with a top surface of two Dirichlet boundary conditions of 80 mm displacement and fixed at the bottom surface. Fig. 17 (b)-(e) shows the comparison of the FP32 and FP64 precision in total displacement and the other three directions on SQUID and FP32 on a local PC. AEPINN can converge under both precisions within 100 s in the total displacement **u**. For $u$ and $v$, AEPINN can converge to a high accuracy of around 200 s. For $w$, since the displacement in this direction is quite small, AEPINN can be overfitted after a quick decreasing stage, meaning the main errors will come from AEPINN in this direction. However, despite the overfitting issue, the L2 relative error of total displacement **u** still maintains a low value after 200 s. With the double precision of FP64, AEPINN can even be faster than it with float 32 precision for about 20 % in the training process. For the $u$ and $v$ direction, FP64 can converge faster in a speedup of 100 % compared with FP32 in this frame problem. Fig. 18 (a)-(d) are the predictions of AEPINN for three directions $u$, $v$, $w$, and the total displacement **u**. Fig. 18 (e)-(h) shows the ABAQUS solver with RTX 3080ti accelerated results with the mesh points of 80172 meaning the mesh size is about 1/43 with 18 s to converge. Fig. 18 (i)-(l) illustrates the differences between the solver and predictions. The

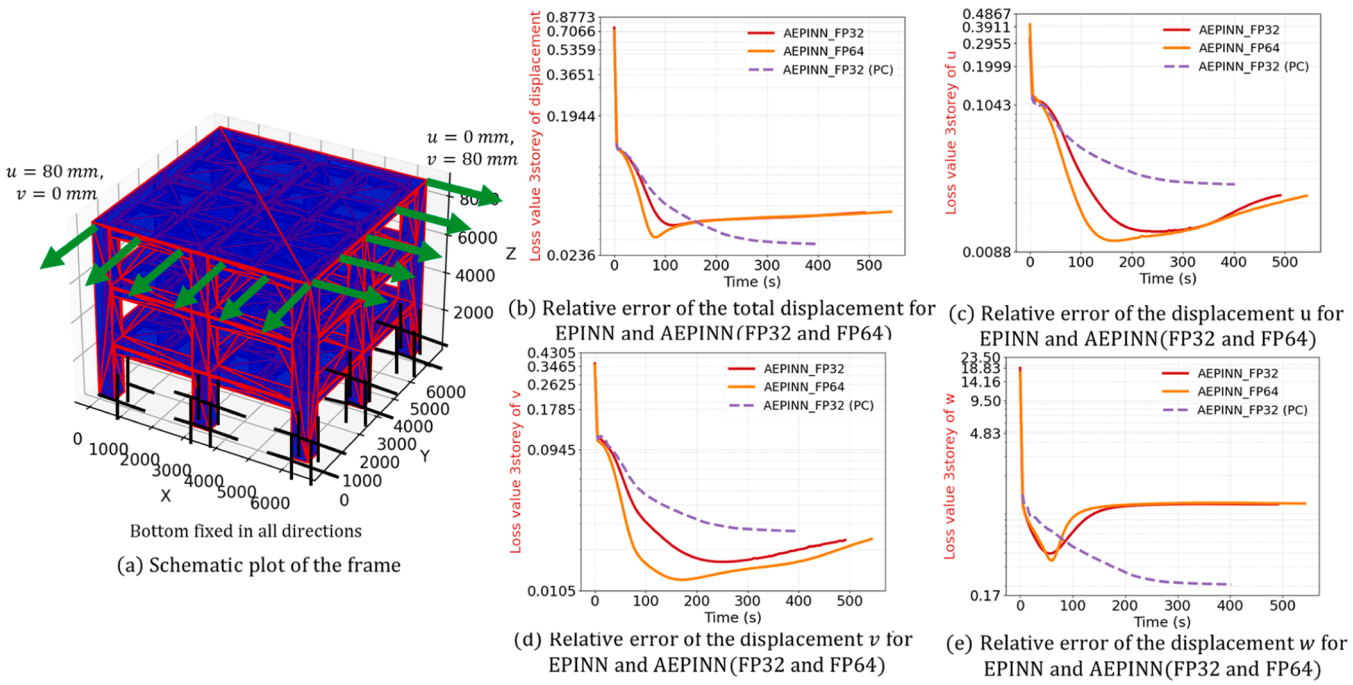



**Fig. 17.** Comparison of the AEPINN performance with FP32 and FP64 precision for simulating 3D frame with two fixed Dirichlet boundary conditions on the top surface.

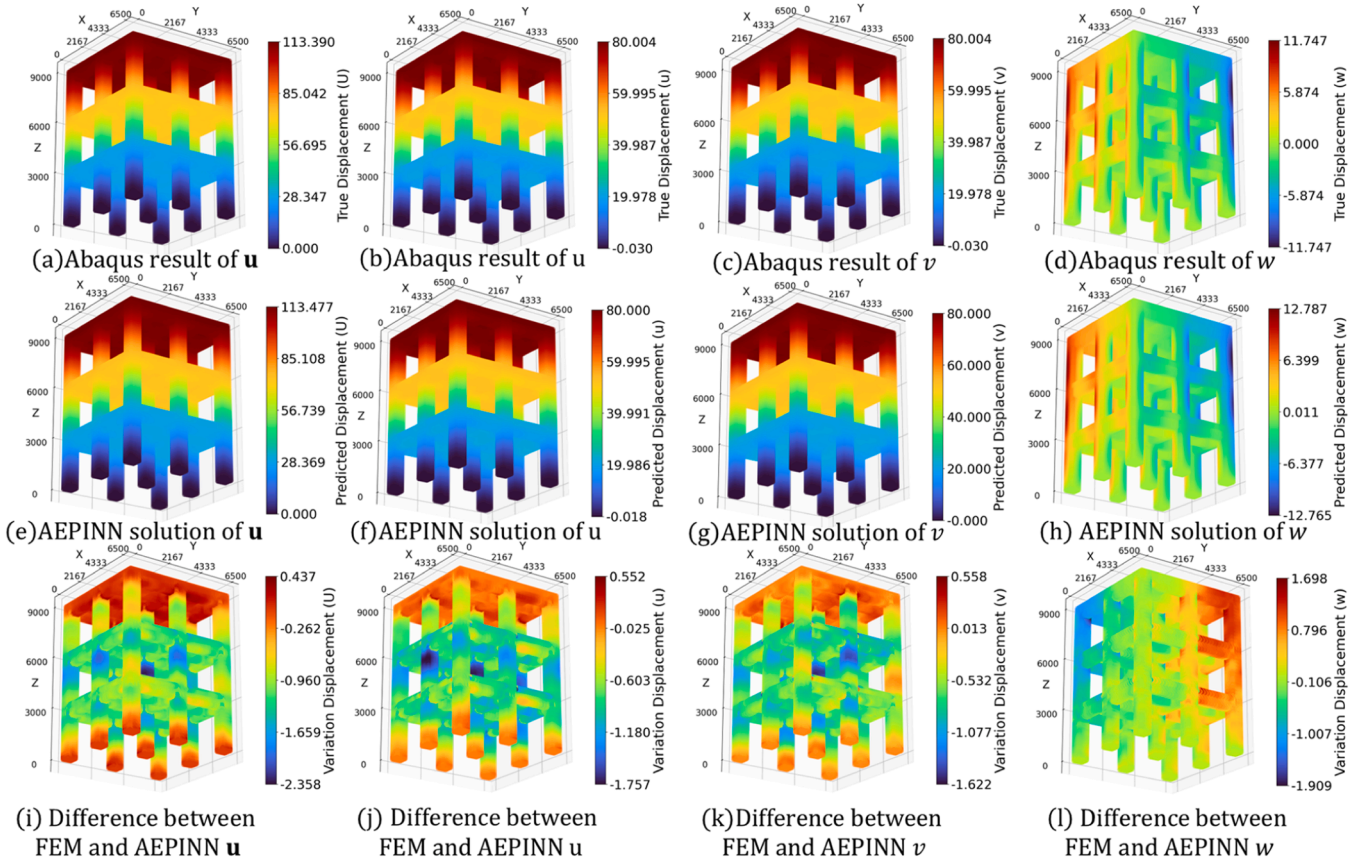



**Fig. 18.** The performance of the AEPINN for simulating 3-storey frame under lateral loading (in units of mm).

average errors of total displacement **u**, displacement in x direction *u*, y direction *v* and z direction *w* are lower than 2.4 mm, 1.76 mm, 1.7 mm, and 1.91 mm, respectively. Considering integrated total displacement error with a value of around 2 %, AEPINN can solve this frame problem within certain error values, which shows the ability of the proposed method AEPINN in solving the static complicated 3D frame problem

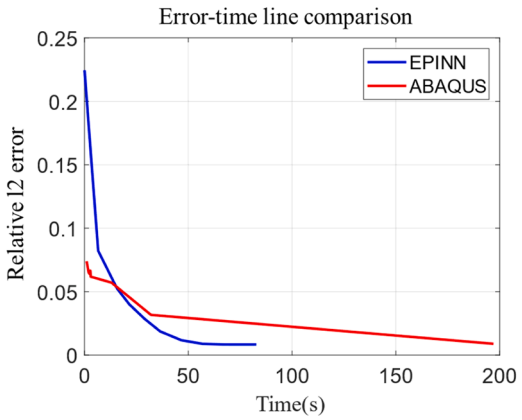


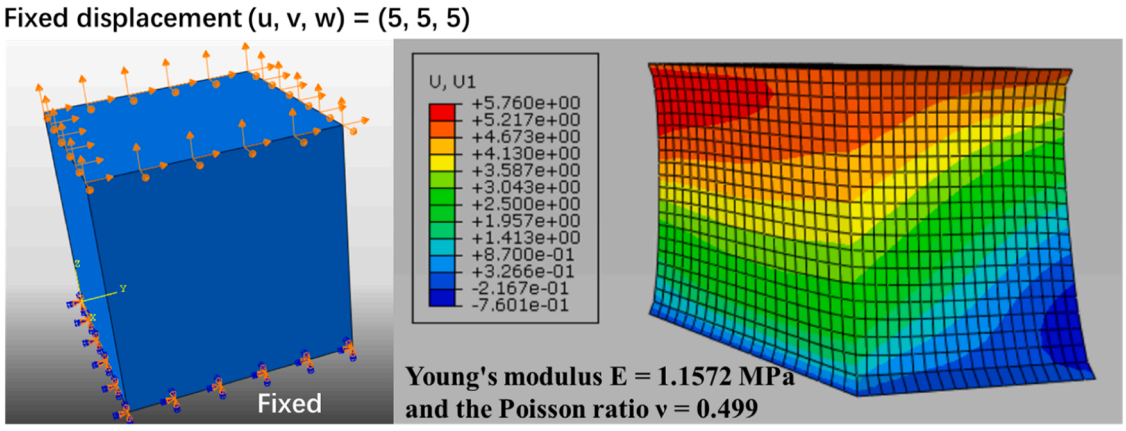

**Fig. 19.** Mesh sensitivity of the frame displacement field comparison between AEPINN and ABAQUS results ranging from 50 mm to 200 mm, take 25 mm ABAQUS case as reference.

displacement solution fields respectively, indicating the potential of AEPINN in applying to more reality cases in the future.

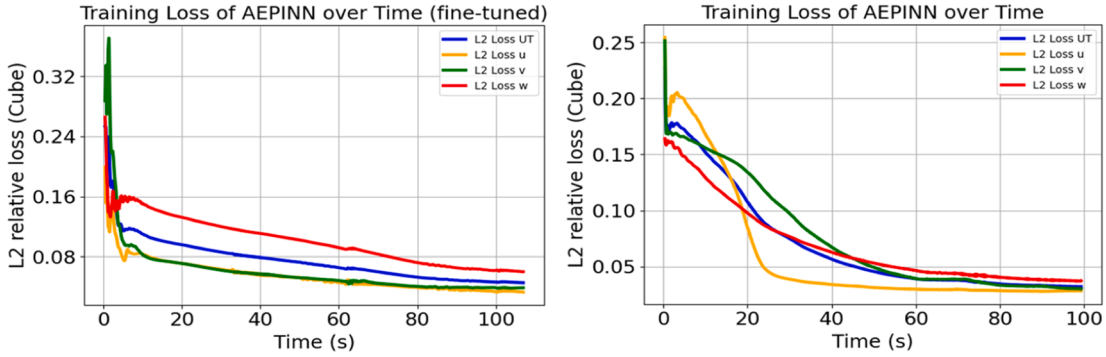
For the elastic static problem, ABAQUS solver with GPU accelerated can obtain solution fields efficiently and effectively. For comparing the efficiency of AEPINN model and FEM in obtaining displacement solution fields with the finest meshed FEM results with 25 mm. We randomly select 30000 fixed sampled points from the input 3D file and train the AEPINN model while validating the model with fine meshed ABAQUS results with 25 mm mesh size. ABAQUS results with other mesh size cases ranging from 50 mm to 200 mm are also compared with the 25 mm case by using the same relative l2 error. Hence, this error-time figure below in Fig. 19 can depict the performance of ABAQUS and AEPINN in approaching the fine mesh size 25 mm case in both accuracy and efficiency. Same as AEPINN, ABAQUS results will be compared with the reference 25 mm case to verify the ability of ABAQUS in solving FEM problems. and other ABAQUS results from mesh size 50 mm (time:197 s, relative l2 error:0.89 %), 75 mm(time:32 s, relative l2 error:3.17 %), 100 mm(time:13 s, relative l2 error:2.37 %), 125 mm(time:3 s, relative l2 error:2.91 %), 150 mm(time:2.8 s, relative l2 error:2.99 %), 175 mm (time:2.4 s, relative l2 error:4.4 %) to 200 mm(time:2 s, relative l2 error:3.5 %) for comparison. Fig. 19 below shows that compared with ABAQUS solver, in the first few seconds, ABAQUS can achieve displacement fields with sparse mesh but with exponential increased time with fine mesh size, while AEPINN can balance the computational efficiency as well as speed to enhance the solution performance compared with 25 mm mesh size ABAQUS case, indicating the potential of AI solution to assist and accelerate finite element analysis in the future. Hence, if we need higher accuracy of displacement fields and satisfied efficiency, AEPINN can be an option in helping solving solid mechanics problems to reduce time costs.

### 3.4. Rubber cube

Although AEPINN can be efficient in solving the linear elastic static solid mechanics problems in 1D, 2D and 3D cases above, nonlinear solid mechanics problems can be great challenges for artificial intelligence frameworks to solve. In our research, to examine the effectiveness of AEPINN in solving nonlinear solid mechanics, a rubber square with length, width, and height of 50 mm and the bottom is fixed, while the top surface is fixed with a displacement value of 5 mm in three directions. As [45,46] examined, the Young's modulus of the rubber square is adopted as the same value $E = 1.1572\text{MP}a$ and the Poisson ratio $\nu = 0.499$ from [46] and Mooney-Rivlin parameters




(a) Schematic plot of the cube rubber example




(b) Training loss of the cube rubber example

**Fig. 20.** Schematic plot of the proposed AEPINN method with optimized hyperparameters in solving nonlinear hyper-elastic rubber problem (in units of mm).

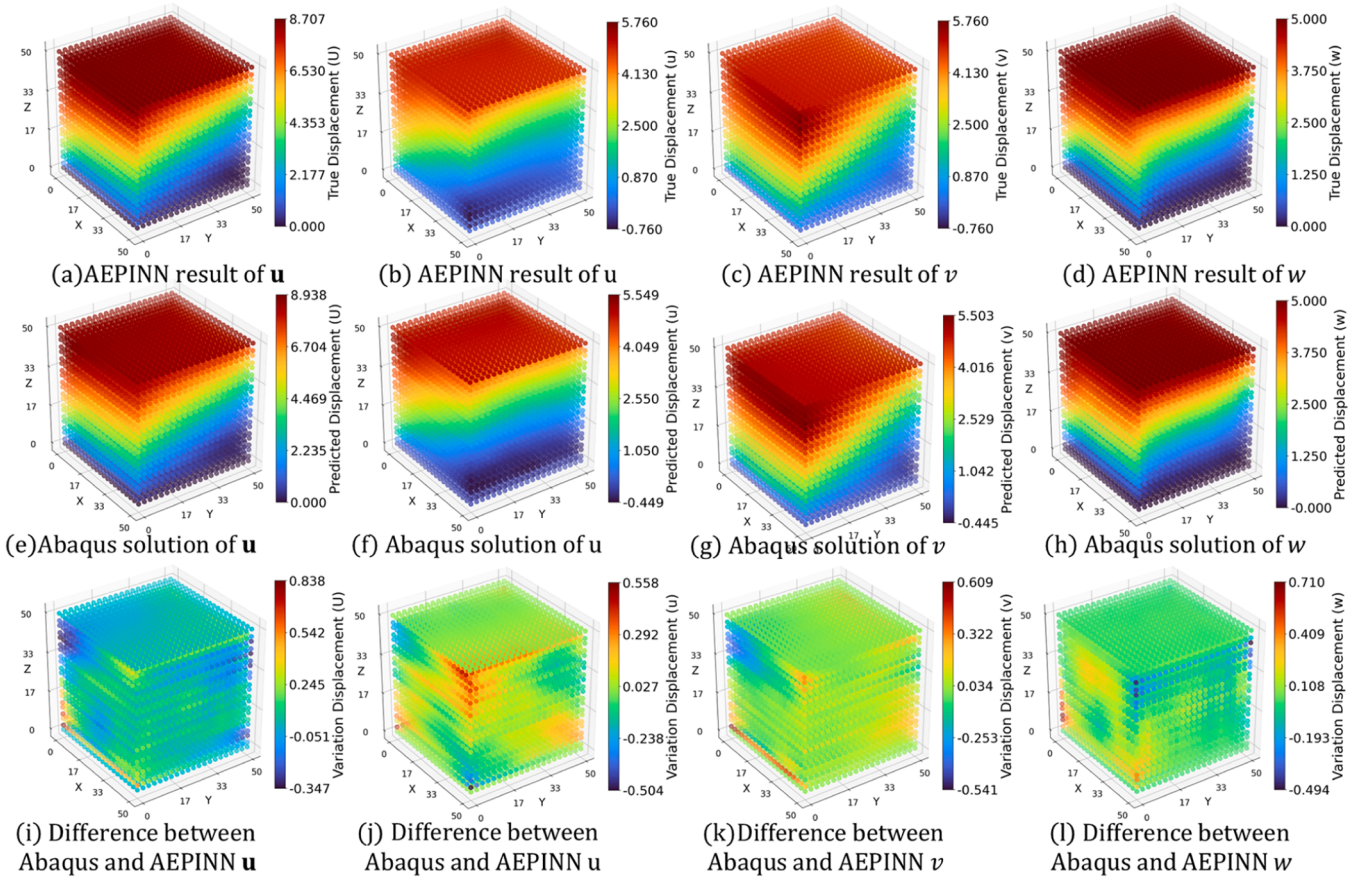



**Fig. 21.** Schematic plot of the proposed AEPINN method with optimized hyperparameters in solving nonlinear hyper-elastic rubber problem (in units of mm).

C10 = 0.006243, C01 = 0.007461. The AEPINN model from case 3.3 frame is fine-tuned from previous trained network configurations and applying the Mooney-Rivlin energy function, AEPINN model can obtain results with relative L2 errors 0.06 in three directions u, *u*, *v* and *w* within 45 s and L2 errors all below 0.05 within 60 s. Fig. 20 shown below can be utilized to illustrate the performance of AEPINN.

Previously, with AutoML optimized hyperparameters, AEPINN can be applied to static elastic problems and solve those faster than conventional PINN and EPINN. In this nonlinear hyper-elastic case, we adopt the previous optimized hyperparameter sets and set TD modes as well as the mesh number equal to the frame case. With optimized hyperparameters, AEPINN can solve this rubber static hyper-elastic within 60 s. Besides, the pre-trained optimized weights of AEPINN in the frame case can be utilized to be fine-tuned for further applications to quick responses towards studied objects. Apparently, with pre-trained AEPINN weights, the model can achieve a speedup compared with directly training AEPINN.

Fig. 21 exhibits the ABAQUS results and AEPINN predicted displacement fields. In all three directions, even in this nonlinear problem, AEPINN can obtain the solution fields with L2 relative errors in all directions lower than 10 % within 100 s. While the previous research papers mentioned didn't indicate the actual training time for solving these nonlinear hyper-elastic problems, the simplest 1D and 2D problems for those methods will require hundreds of seconds to solve the static elastic problems instead of hyper-elastic problems. Hence, compared with those methods, AEPINN is compatible with artificial intelligence models in solving displacement solution fields. Of course, in this cube rubber case, the stress fields are not listed for currently the AEPINN hasn't directly output stress fields and the derived stress fields from displacement data will induce a series of errors due to the numerical errors in rounding numbers and calculating derivatives. In the future, we will analyze the AEPINN to obtain full fields of problems.

### 3.5. *Qualitative analysis and discussion of the AEPINN process*

One of the most important things to consider the qualitative analysis of the proposed method is to measure the uncertainty compared to current research. However, many studies have conducted related validation without analyzing the model performance corresponding to the automated machine-learning iterations. In this part, we conduct the PCA of the bracket and frame cases to analyze the impacts of different hyperparameters towards the performance of the proposed AEPINN model in solving purely physics-informed solid mechanics problems. Since this analysis is more about the performance of this framework, the irrelevant hyperparameter decay step is removed. Fig. 22 denotes the PCA contour results of the AutoML process. Fig. 22 (a) and (c) display how the proposed method can assist the AutoML process in discovering the regions where hyperparameter sets are effective for the model to solve problems. Those internal areas are surrounded by low efficiency boundaries where all those sets are insufficient for AEPINN to predict the displacement fields successfully. In AutoML results, the loss values lower than 0.5 are near the centre of the PCA geometry, denoting the performance under uncertain hyperparameters near the centre sets will be able to approach solution fields respectively. Fig. 22 (b) and (c) show that the loadings in PCA 1 almost all come from the sampling points, which is reasonable since sampling is the data resource. For PCA 2, loadings about mesh number and modes appear different trends for two problems. The learning rate did not contribute a lot to the final convergence in these two cases.

Although the proposed AEPINN methods can solve the static frame case and nonlinear hyperelastic cube rubber case, our research still requires further development in improving stress prediction so that the

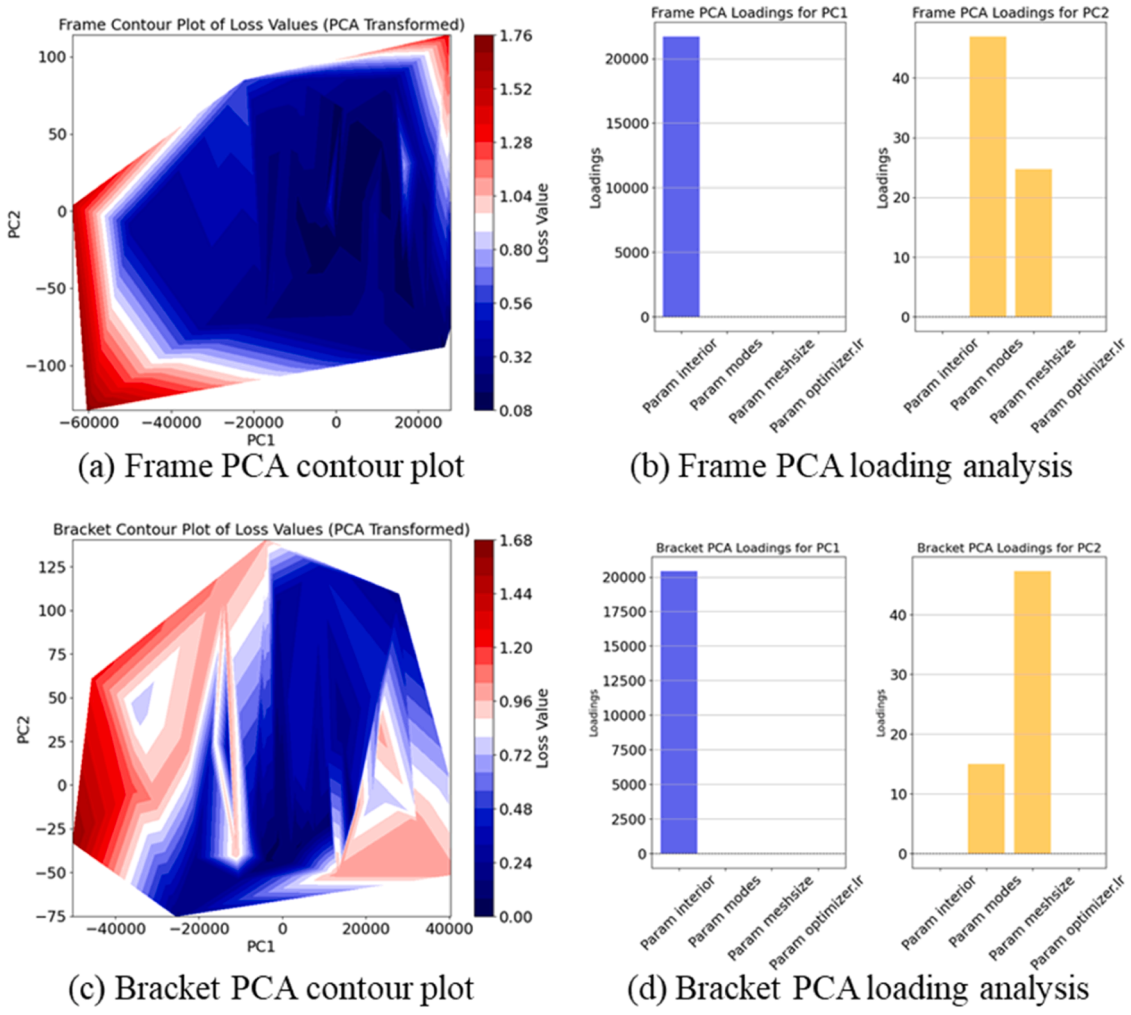



**Fig. 22.** PCA contour results and loadings analysis of hyperparameters.

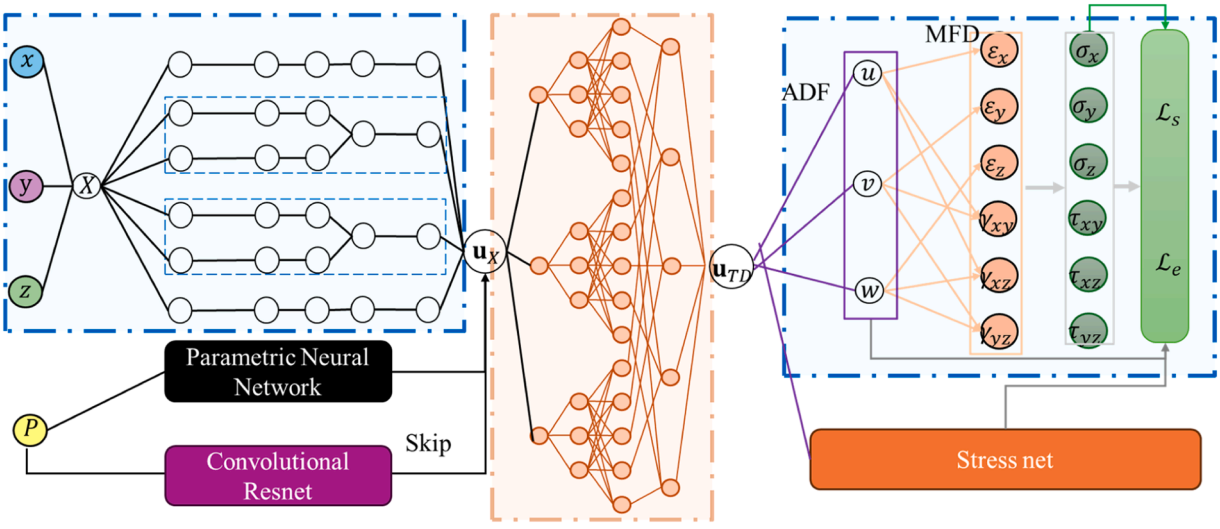



**Fig. 23.** Strain fields in three normal direction comparison between Abaqus and AEPINN results.

proposed methodology can be applied to real structures. For the frame case, the top and bottom parts are fixed with boundary conditions which are applied in AEPINN model. However, the conducted Dirichlet boundary condition cannot fully constrain the displacement in other parts without any displacement errors. Therefore, strain and stress, the derivatives derived from displacement fields may hold some errors due to the accumulated errors. One possible solution is to predict the displacement fields and stress fields simultaneously and output these

two types of results, which could help the model avoid the accumulated errors due to derivatives. Besides, in this model, only Dirichlet boundary conditions are applied and Neuman boundary conditions are not considered since we are focusing on the displacement solution fields. In the future, these two methods can be considered to extend our proposed AEPINN in solving displacement, strain and stress fields efficiently.

Another future direction is to consider the material parameters and geometric information during the automated machine-learning process. In this study, AutoML did not consider various material and geometry parameters of studied objects such as different Young's Modulus values, Poisson ratio values, and the geometry setting such as the length, width, height, and thickness of the cube. Current AutoML cannot consider these parameters since the proposed APEINN only be designed for the specific material and the geometric input information from other cases cannot be embedded. To further extend our model, the tensor decomposition part can be developed to include the required information. Fig. 23 displays the further development of the AEPINN framework. The external parameters will first pass through the parametric net to preprocess the information so that the results can be embedded into the initial state with the shape function. Secondly, the convolutional Resnet with fewer trainable parameters to skip the parameters if those weights cannot provide useful information after the parametric net. Finally, the stress net will be included in the third part which will be trained to predict the stress fields while the displacement fields and stress can be verified with each other in the final loss function.

## 4. Conclusions

The proposed AEPINN framework is developed to improve the performance of EPINN in solving solid mechanical problems, which adopts the Bayesian optimization TPE framework as the basis and the principles of the hyperparameter optimization process. This study is focused on an extreme case of zero-shot learning, where we only apply the governing equation without any data. It is for the first time we find this PINN architecture can be accelerated to the same level or even faster than conventional FE solver, which has already been optimized and developed for more than 50 years so far. In more practical case such as inverse problems, structural health monitoring problems and structural design optimization problem, when we have additional dataset from test or FE simulation, we believe the proposed AEPINN architecture can reach significant speedup compared to conventional FE simulation, as this is the primary objective of Scientific Machine Learning algorithms.

Because this AEPINN framework adopts the EPINN architecture, the training process can converge without any labeled data of the solution field. The AEPINN framework showed its ability to solve solid mechanical problems directly without any data. The AEPINN framework is also discussed to explore complex solid mechanical problems for complicated shaped objects. The major contributions of this paper can be concluded as follows:

(1) The AutoML EPINN framework combines an automated machine-learning method that utilizes the BO-TPE to optimize the hyperparameters and architecture of the adopted PINN and EPINN. With the matching principle for shape function or interpolation-based framework, automated machine learning can achieve an obvious speedup.
(2) The developed AEPINN is a mesh-free framework with 3D design geometry or scanned information as input to consider irregularly shaped objects by sampling points randomly within the interior and boundary parts to evaluate the irregularly shaped 3D static solid mechanics problems, which explores the possibility of mesh-free methodologies in solving solution fields of solid mechanics.
(3) For the 2D plane stress case, AEPINN can achieve 20 times speedup compared with the EPINN architecture, similar speed compared with ABAQUS solver with GPU accelerated, and more than 200 times compared with conventional PINN.
(4) For the 3D bracket traction problem, AEPINN can converge quickly within 2 minutes, achieving four times speedup compared with EPINN architecture, Five times the speedup of the Matlab FEM solver, and more than 400 times the speedup of the conventional PINN framework. can achieve a speedup of two times compared with EPINN architecture and 200 times compared with conventional PINN. Considering the applicability of physics-informed architecture, this study conducts the 3D complex 3-story frame model for the AEPINN framework to solve. AEPINN can solve this problem within 2 minutes and can capture the displacement fields within 400 seconds, displaying the potential of this AEPINN framework to contribute to addressing solid mechanics with AI in the future.
(5) The optimized hyperparameters from Automated machine learning can be transferred to other solid mechanics problems without rerunning the AutoML process for every problem. The optimized hyperparameters from the frame case are adopted to solve the nonlinear hyperelastic cube rubber problem which cannot be solved unless Abaqus utilizes the hybrid element as well as the nonlinear geometry methods. The optimized AEPINN can solve the displacement solutions fields of hyperelastic problems within 60 s to reach the relative l2 error below 0.1 which is comparably fast compared with existing, exhibiting the potential of the proposed methodology in applying to real structures.

## CRediT authorship contribution statement

**Zhu Yingjie:** Resources. **Deng Xiaowei:** Writing – review & editing, Supervision. **Tian Xiaoge:** Writing – original draft, Visualization, Validation, Methodology, Investigation. **Kim Chul-Woo:** Writing – review & editing, Software, Resources. **Wang Jiaji:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Methodology, Data curation, Conceptualization.

## 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

The work in this paper is financially supported by the National Natural Science Foundation of China Project 52408221, Hong Kong Innovation and Technology Support Programme (Mid-stream, theme-based, ITS/041/23MX). The authors would like to thank National Supercomputer Center in Guangzhou for providing high performance computational resources. The findings in the paper reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the sponsor.

## Data availability

Data will be made available on request.

## References

[1] Zienkiewicz OC, Taylor RL, Zhu JZ. The finite element method: its basis and fundamentals. Elsevier; 2005.
[2] Schneider T, et al. A large-scale comparison of tetrahedral and hexahedral elements for solving elliptic PDEs with the finite element method. ACM Trans Graph (TOG) 2022;41(3):1–14.

[3] Kang F, et al. Multi-parameter inverse analysis of concrete dams using kernel extreme learning machines-based response surface model. Eng Struct 2022;256:113999.
[4] Jin H, Zhang E, Espinosa HD. Recent advances and applications of machine learning in experimental solid mechanics: a review. Appl Mech Rev 2023;75(6):061001.
[5] Samaniego E, et al. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications. Comput Methods Appl Mech Eng 2020;362:112790.
[6] Jagtap AD, Kharazmi E, Karniadakis GE. Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. Comput Methods Appl Mech Eng 2020;365:113028.
[7] Wen W, Zhang C, Zhai C. Rapid seismic response prediction of RC frames based on deep learning and limited building information. Eng Struct 2022;267:114638.
[8] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[9] Zhang E, et al. Analyses of internal structures and defects in materials using physics-informed neural networks. Sci Adv 2022;8(7):eabk0644.
[10] Haghighat E, et al. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Comput Methods Appl Mech Eng 2021;379:113741.
[11] Liu W-H, Zhang L-W, Dai J-G. A physics-informed and data-enhanced tensile stress-strain model for UHPFRC. Eng Struct 2023;285:115989.
[12] Karniadakis GE, et al. Physics-informed machine learning. Nat Rev Phys 2021;3(6):422–40.
[13] Diao Y, et al. Solving multi-material problems in solid mechanics using physics-informed neural networks based on domain decomposition technology. Comput Methods Appl Mech Eng 2023;413:116120.
[14] Jeong H, et al. A physics-informed neural network-based topology optimization (PINNTO) framework for structural optimization. Eng Struct 2023;278:115484.
[15] Li G, et al. Static analysis of two-side supported 2-ply laminated glass panes through physics-informed neural networks. Eng Struct 2024;309:118038.
[16] Saha S, et al. Hierarchical deep learning neural network (HiDeNN): an artificial intelligence (AI) framework for computational science and engineering. Comput Methods Appl Mech Eng 2021;373:113452.
[17] Zhang L, et al. HiDeNN-TD: reduced-order hierarchical deep learning neural networks. Comput Methods Appl Mech Eng 2022;389:114414.
[18] Li H, et al. Convolution hierarchical deep-learning neural network tensor decomposition (C-HiDeNN-TD) for high-resolution topology optimization. Comput Mech 2023;72(2):363–82.
[19] Park C, et al. Convolution hierarchical deep-learning neural network (c-hidenn) with graphics processing unit (gpu) acceleration. Comput Mech 2023;72(2):383–409.
[20] Wang J, et al. Exact dirichlet boundary physics-informed neural network EPINN for solid mechanics. Comput Methods Appl Mech Eng 2023;414:116184.
[21] Sukumar N, Srivastava A. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. Comput Methods Appl Mech Eng 2022;389:114333.
[22] Andonie R. Hyperparameter optimization in learning systems. J Membr Comput 2019;1(4):279–91.
[23] Escapil-Inchauspé P, Ruz GA. Hyper-parameter tuning of physics-informed neural networks: Application to Helmholtz problems. Neurocomputing 2023;561:126826.
[24] Chen Y, et al. Physics-Informed LSTM hyperparameters selection for gearbox fault detection. Mech Syst Signal Process 2022;171:108907.
[25] Zhang Q, et al. TBM performance prediction with Bayesian optimization and automated machine learning. Tunn Undergr Space Technol 2020;103:103493.
[26] Lindauer M, et al. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. J Mach Learn Res 2022;23(54):1–9.
[27] Ezati M, Esmaeilbeigi M, Kamandi A. Novel approaches for hyper-parameter tuning of physics-informed Gaussian processes: application to parametric PDEs. Eng Comput 2024:1–20.
[28] Watanabe, S., Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. arXiv preprint arXiv:2304.11127, 2023.
[29] Nguyen H-P, Liu J, Zio E. A long-term prediction approach based on long short-term memory neural networks with automatic parameter optimization by Tree-structured Parzen Estimator and applied to time-series data of NPP steam generators. Appl Soft Comput 2020;89:106116.
[30] Ozaki, Y., et al. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. in *Proceedings of the 2020 genetic and evolutionary computation conference*. 2020.
[31] Mao J, et al. Automated Bayesian operational modal analysis of the long-span bridge using machine-learning algorithms. Eng Struct 2023;289:116336.
[32] Zhang L, et al. Hierarchical deep-learning neural networks: finite elements and beyond. Comput Mech 2021;67:207–30.
[33] Elshawi, R., M. Maher, and S. Sakr, *Automated machine learning: State-of-the-art and open challenges*. arXiv preprint arXiv:1906.02287, 2019.
[34] Thornton, C., et al. *Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms*. in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. 2013.
[35] Wang S, Wang H, Perdikaris P. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. Comput Methods Appl Mech Eng 2021;384:113938.
[36] Palar, P.S., et al. *On the use of surrogate models in engineering design optimization and exploration: The key issues*. in Proceedings of the genetic and evolutionary computation conference companion. 2019.
[37] Snoek J, Larochelle H, Adams RP. Practical bayesian optimization of machine learning algorithms. Adv Neural Inf Process Syst 2012;25.
[38] Wilson J, Hutter F, Deisenroth M. Maximizing acquisition functions for Bayesian optimization. Adv Neural Inf Process Syst 2018:**31**.
[39] He X, Zhao K, Chu X. AutoML: a survey of the state-of-the-art. Knowl-Based Syst 2021;212:106622.
[40] Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Rev 2009;51(3):455–500.
[41] Tang K, Wan X, Yang C. DAS-PINNs: a deep adaptive sampling method for solving high-dimensional partial differential equations. J Comput Phys 2023;476:111868.
[42] Nvidia. ⟨https://docs.nvidia.com/deeplearning/modulus/index.html⟩. 2024.
[43] Nwankpa, C., et al., Activation functions: Comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378, 2018.
[44] Rao C, Sun H, Liu Y. Physics-informed deep learning for computational elastodynamics without labeled data. J Eng Mech 2021;147(8):04021043.
[45] Goodbrake C, Motiwale S, Sacks MS. A neural network finite element method for contact mechanics. Comput Methods Appl Mech Eng 2024;419:116671.
[46] Bai J, et al. A robust radial point interpolation method empowered with neural network solvers (rpim-nns) for nonlinear solid mechanics. Comput Methods Appl Mech Eng 2024;429:117159.