Article

Optimizing Hyperparameters and Architecture of Deep Energy Method

Charul Chadha[1], Diab Abueidda[1,2*], Seid Koric[1,2], Erman Guleryuz[2] and Iwona Jasiuk[1]

[1.] Department of Mechanical Science and Engineering, the University of Illinois at Urbana-Champaign, IL, USA
[2.] National Center for Supercomputing Applications, the University of Illinois at Urbana-Champaign, IL, USA

Abstract:
The deep energy method (DEM) employs the principle of minimum potential energy to train neural network models to predict displacement at a state of equilibrium under given boundary conditions. The accuracy of the model is contingent upon choosing appropriate hyperparameters. The hyperparameters have traditionally been chosen based on literature or through manual iterations. The displacements predicted using hyperparameters suggested in the literature do not ensure the minimum potential energy of the system. Additionally, they do not necessarily generalize to different load cases. Selecting hyperparameters through manual trial and error and grid search algorithms can be highly time-consuming. We propose a systematic approach using the Bayesian optimization algorithms and random search to identify optimal values for these parameters. Seven hyperparameters are optimized to obtain the minimum potential energy of the system under compression, tension, and bending loads cases. In addition to Bayesian optimization, Fourier feature mapping is also introduced to improve accuracy. The models trained using optimal hyperparameters and Fourier feature mapping could accurately predict deflections compared to finite element analysis for linear elastic materials. The deflections obtained for tension and compression load cases are found to be more sensitive to values of hyperparameters compared to bending. The approach can be easily extended to 3D and other material models.

Keywords: Elasticity; Machine learning; Minimum potential energy; Partial differential equations (PDEs); Physics-informed neural network

1. Introduction

With the advancement in computational technology and complexity in design, engineers seek to predict the component's performance before manufacturing it. These predictions involve complex boundary value problems (BVP), which require solutions to partial differential equations (PDEs). Typically, these PDEs are solved through numerical approximations. In solid mechanics, conventional methods to solve PDEs include mesh-free methods[1,2], finite element analysis (FEA)[3], and isogeometric analysis[4,5]. However, these methods are posed with challenges such as obtaining robust and accurate solutions for ill-posed, high-dimensional, or coupled problems, to name a few.
Machine learning methods are rapidly developing as an alternative to traditional approaches to overcome the issues mentioned above. Machine learning methods can be classified as data-driven models[6–16] and physics-informed neural networks (PINNs)[17–24]. In data-driven models, data from experimental and computational results are used to train the models. Although this method can capture complex physical phenomena, the amount of data required to train the models, data quality, and data generation process impede its widespread implementation[25]. Another application of data-driven models in mechanics is to provide an automated framework for predicting constitutive models for material behavior, as discussed by Flaschel et al.[26] , Yang et al. [27] and Chen[28].
Alternatively, PINNs with automatic differentiation were first introduced by Raisi et al.[17]. They used a deep neural network (DNN) to train the model based on physical laws at random points in the domain. PINNs hold several advantages over data-driven models as they do not necessarily require data labeling. Even if data are used in addition to the physical laws, PINNs do not need large quantities of datasets compared to entirely data-driven models. Two types of PINNs models are rapidly developing in solid mechanics: deep collocation method (DCM)[22–24,29] and deep energy methods (DEM)[30–32]. In DCM, residuals of strong form define the loss function at points sampled from the physical domain, corresponding boundary, and initial conditions (collocation points).

* *Corresponding author: Diab Abueidda*
E-mail address: abueidd2@illinois.edu

This method, introduced by Raisi et al.[17], has been extended and improved by researchers[33,34]. Haghighat et al. have used a PINN technique to solve the equations of coupled flow and mechanical deformation in porous media for single-phase and multiphase flows[35]. Though the DCM has been widely successful in predicting outcomes, the approach requires higher-order gradients than the DEM.
Exploring applications of deep neural networks (DNN), Weinan and Bing[36] developed the deep ritz method to solve variational problems. Nguyen-Thanh et al.[31] expanded the work proposed by Weinan and Bing to develop DEM. In DEM, the system's potential energy, expressed as a loss function, is minimized to predict the system's displacements. The DEM method is suited for problems where an energy functional exists and reduces dependencies on PDEs of the base function. Implementation of DEM for several BVPs has been reported in the literature [30,32,37]. Extending this work, Fugh et al.[38] demonstrated that DEM and DCM fail to accurately resolve displacement and stresses at stress concentration regions. To overcome this issue, they proposed modified DEM (mDEM), in which they added stresses to the loss function. The modification allowed the successful resolution of stresses around the stress concentration regions. Abueidda et al.[39] enhanced the model by developing PINN that combines residuals of the strong form and the system's potential energy. The proposed formulation yielded a loss function with many loss terms. As a result, the coefficient of variation weighting scheme was also introduced in the loss function to assign the weight dynamically and adaptively for each term.
The accuracy of the DNN-based algorithms described above depends on the hyperparameters selected for the model. These hyperparameters define the architecture of the DNN and affect the weights and biases obtained while training the models. The optimal values of hyperparameters are problem-specific and may not be the same for different BVPs. Due to this, recent results demonstrate that challenges in the optimization of hyperparameters in large and multi-layered models impede scientific progress[40]. Tuning these hyperparameters through manual trial and error is time-consuming. As a result, researchers have employed grid search, random search, and optimization algorithms to obtain best combination of hyperparameters [21,40–43]. Still, the studies on optimizing hyperparameters and architectures of PINNs for solving mechanics problems remain uncommon.
This paper utilizes DEM and develops a systematic approach using Bayesian optimization algorithms and random search to identify optimal values for hyperparameters when using the PINN method. A two-loop framework is employed for the hyperparameter optimization process to search a non-convex optimization space containing both discrete and continuous variables.
Also, Fourier feature mapping was employed to improve the model's accuracy. Tancik et al.[44] showed that passing input points through a simple Fourier feature mapping enables a multilayer perceptron (MLP) to learn high-frequency functions in low-dimensional problem domains. However, based on the neural tangent kernel (NTK) theory, they have indicated that a standard MLP fails to learn high frequencies. Therefore, they have employed a Fourier feature mapping to mitigate this spectral bias to transform the effective NTK into a stationary kernel with a tunable bandwidth. The modifications done in our paper demonstrate significant improvement in displacement and strain distribution obtained for 2D load cases for elastic materials compared to results obtained when hyperparameters specified in the literature were used. Though the current study focuses on 2D- plane stress elastic materials, the approach can easily be extended to 3D and to materials with non-elastic material properties (e.g., hyperelastic materials)
The paper is divided into six sections. Section 2 details governing equations for BVPs and DEM. The modifications to the DEM process to improve its performance are described in Section 3. Section 4 compares the hyperparameters obtained for three different load cases (uniform compression, uniform tension, and uniform bending). The best hyperparameters obtained in Section 4 are used to solve numerical problems involving different geometry, partial loading, and 2D cross-section with a hole in the geometry in Section 5. Finally, we conclude the paper in Section 6 by stating important observations and highlighting possible future work directions.

2. Mathematical formulation

2.1 Boundary value problem

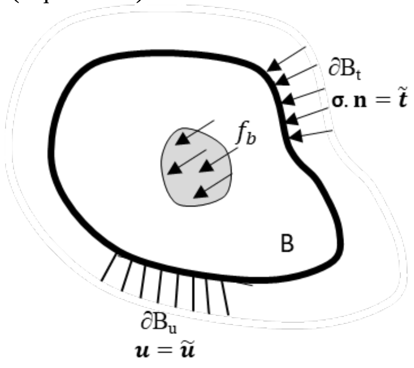
Let us consider an elastic body B $\subset \mathbb{R}^2$ bounded by the boundary ∂B in the initial configuration, as shown in Figure 1. The body is subjected to the Dirichlet boundary and Neumann boundary conditions on boundaries ∂B_u and ∂B_t, respectively, such that $\partial B_u \cup \partial B_t = \partial B$ and $\partial B_u \cap \partial B_t = \emptyset$. Due to the applied boundary conditions, the body undergoes deformations which can be found using the following governing equations, assuming negligible inertial forces:

$$\begin{aligned} &\text{Equilibrium:} && \nabla_x \boldsymbol{\sigma} + \boldsymbol{f}_b = 0 && in\ B \\ &\text{Dirichlet boundary:} && \mathbf{u} = \tilde{\boldsymbol{u}} && \text{on } \partial B_u \\ &\text{Neumann boundary:} && \boldsymbol{\sigma}.\boldsymbol{n} = \tilde{\boldsymbol{t}} && \text{on } \partial B_t \end{aligned} \quad (3)$$

where $\boldsymbol{\sigma}$ and $\boldsymbol{f}_b$ denote Cauchy stress tensor and body force, respectively. $\boldsymbol{n}$ is the outward normal along the boundary B_t, $\boldsymbol{u}$ is the displacement field, and $\tilde{\boldsymbol{u}}$ is displacement prescribed on ∂B_u. In the case of linear elastic materials, the Cauchy stress tensor is related to the linear strain tensor using Hooke's law (equation 4).

$$\sigma_{ij} = C_{ijkl}\epsilon_{kl} \quad (4)$$

where C_{ijkl} is the fourth-order stiffness tensor. The stiffness tensor for linear elastic material can be calculated using elastic material's strain energy density function (ψ) (Equation 5).



∂B_t
$\boldsymbol{\sigma}.\mathbf{n} = \tilde{\boldsymbol{t}}$
f_b
B
∂B_u
$\boldsymbol{u} = \tilde{\boldsymbol{u}}$

Figure 1: Solid body with boundary conditions.

$$\psi(\boldsymbol{\epsilon}) = \mu(\boldsymbol{\epsilon}:\boldsymbol{\epsilon}) + \frac{1}{2}\lambda(\boldsymbol{\epsilon}:1)^2 \quad (5)$$

$$\boldsymbol{C} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} = \frac{\partial^2 \psi(\boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon} \partial \boldsymbol{\epsilon}} \quad (6)$$

$$\boldsymbol{\sigma} = \frac{\partial \psi(\boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}} = 2\mu\boldsymbol{\epsilon} + \lambda tr(\boldsymbol{\epsilon})\boldsymbol{I}$$

In the above equations, μ and λ are the Lame's constants. The strain tensor (small deformations) relates to displacements by: (7)

$$\boldsymbol{\epsilon} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)$$

In the case of 2D plane stress condition, the relation between Cauchy stress tensor and strain tensor can be reduced to:

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{Bmatrix} = \begin{bmatrix} 2\mu + \lambda & \lambda & \lambda \\ \lambda & 2\mu + \lambda & \lambda \\ 0 & 0 & 2\mu \end{bmatrix} \begin{Bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{xy} \end{Bmatrix} \quad (8)$$

2.2 Principle of minimum potential energy

The boundary value problem stated above is the strong form of the governing equations, which can be challenging to solve. As a result, a weak form based on the principle of minimum potential energy is often employed. According to the principle of minimum potential energy, out of all the possible kinematically admissible deformations, a conservative structural system assumes displacements that minimize total potential energy at equilibrium [45]. For a linear elastic body (shown in Figure 1), the total energy of the system can be expressed as:

$$\Pi = U - W = \int_B \psi(\boldsymbol{\epsilon})dV - \int_B \boldsymbol{f}_b.\boldsymbol{v}\, dV - \int_{\partial B_t} \tilde{\boldsymbol{t}}.\boldsymbol{v}\, dA \quad (9)$$

where U is the system's internal energy, W is the energy due to externally applied load, and $\boldsymbol{v}$ is a kinematically admissible displacement field. The total potential energy is treated as the loss function ($\mathcal{L}$) in DEM, with an objective to obtain $\boldsymbol{v}$ such that the system's potential energy is minimum.

$$\mathcal{L}(\boldsymbol{v}) = \min_{\boldsymbol{v}\in H} \Pi(\boldsymbol{v}) = \int_B \psi(\boldsymbol{\epsilon})dV - \int_B \boldsymbol{f}_b.\boldsymbol{v}\, dV - \int_{\partial B_t} \tilde{\boldsymbol{t}}.\boldsymbol{v}\, dA \quad (10)$$

where H is the space of admissible functions. The Gauss-quadrature integration rule (described in Appendix A) was employed to calculate integrations listed in Equation 10.

2.3 Architecture for DEM

The DEM consists of a feedforward artificial neural network (ANN), as shown in Figure 2. ANN can be used as universal approximators for a number of functions [46,47]. In general, ANN consists of sets of neurons arranged in layers. Each neuron is connected to all the neurons in adjacent layers. The number of layers (N) and neurons in each layer are predefined according to specific applications. In DEM, the number of neurons in the first and last layer (also known as input and output layers, respectively) is determined by the system's dimension (two in the case of 2D and three in 3D). The programmer predefines the number of neurons in the layers between the input and output layers (also called hidden layers).

Mathematically, the value of each neuron connecting the previous layer (n_{l-1}) is calculated using a linear combination of weights (**w**) and biases (**b**), as shown in Equation 11. An activation function (φ: ℝ→ ℝ) then operates on the calculated value to determine the value passed by the neuron.

$$\widehat{\boldsymbol{z}}^l = \varphi(\boldsymbol{w}^l \boldsymbol{z}^{l-1} + \boldsymbol{b}^l) \tag{11}$$

The values of weights and biases are determined such that the loss function ($\mathcal{L}\left(\widehat{z}^l(w,b)\right)$ or $\mathcal{L}(\Theta)$) is minimized. In a so-called backpropagation procedure, an optimization algorithm can be used to iterate over the loss function to predict values of weights and biases such that:

$$\boldsymbol{\Theta}^* = arg_{\Theta}\ min\ \mathcal{L}(\boldsymbol{\Theta}) \tag{12}$$

Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm is used in the examples shown in the following sections to minimize the loss function and demonstrate the performance of DEM. However, other algorithms can also be used. The backpropagation capabilities of PyTorch are used to calculate gradients of the loss function.

In the above description, parameters like the number of layers and number of neurons in each layer need to be defined before ANN can be trained. These parameters, called hyperparameters, define the architecture of the ANN and control the learning process of the model. It is essential to obtain optimal values for the hyperparameters to obtain a minimum value for the loss function. The number of hyperparameters and their values vary according to the given problem statement. Thus, it can be time-consuming to predict optimal values for these parameters. Six hyperparameters based on the architecture of DEM were determined. These are the total number of layers, number of neurons in hidden layers, activation function, standard deviation in predicting weights and biases, learning rate for the L-BFGS algorithm, and the number of epochs for the L-BFGS algorithm to achieve convergence.

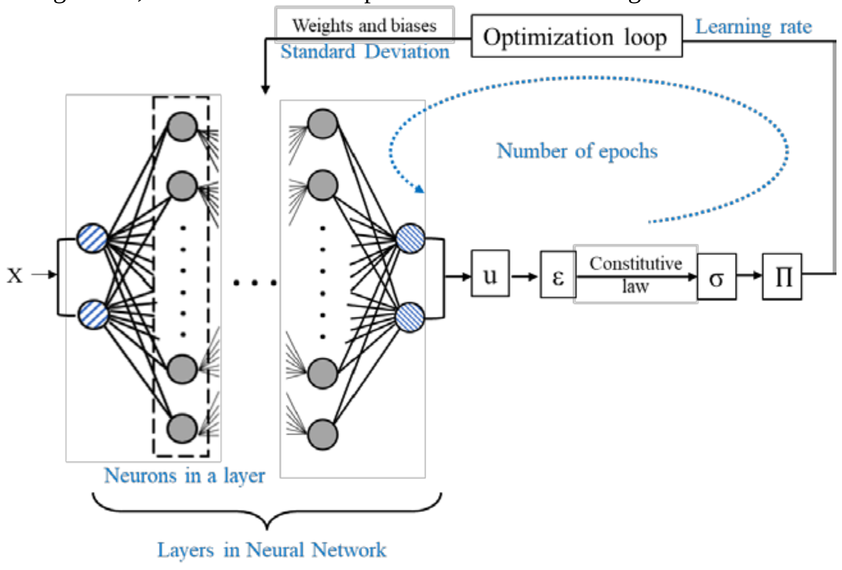


Figure 2: ANN for DEM (Hyperparameters marked in blue).

3. Modifications to the deep energy method

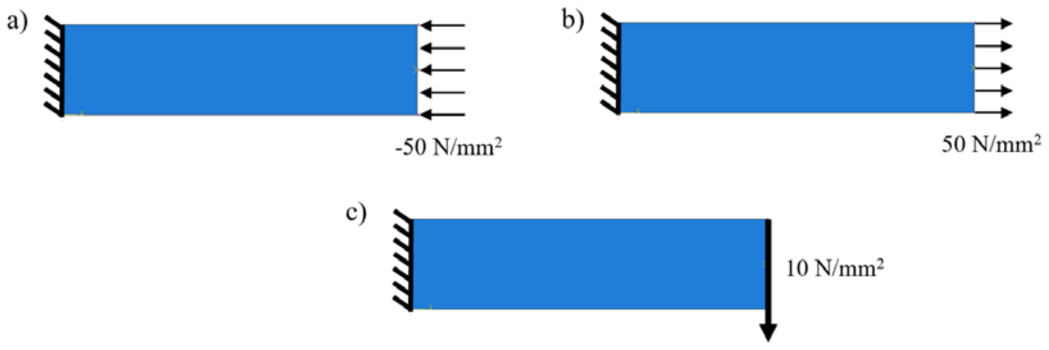


Figure 3: The three load cases: (a) uniform compression, (b) uniform tension, (c) and bending.

We start by implementing the DEM discussed in the work of Nguyan et al.[25]. The DEM model was developed to predict deflections in a 2D elastic rectangular plate under plane stress conditions. An elastic plate of dimensions 4x1 mm^2, with Young's modulus of 1000 MPa, and the Poisson's ratio of 0.3, was fixed at one end. Compression, tension, and bending loads were applied on the other end of the plate, as shown in Figure 3. The predicted deflections in the plate under the three load cases for plane stress are shown in

Appendix B. We noticed that though displacements predicted by DEM closely matched with displacements obtained from FEM along the x-axis, a significant loss in symmetry was observed in predicted displacements along the y-axis, under tension and compression. This deviation also resulted in inaccurate predictions of strain.

Three areas of improvement were identified to overcome this issue and improve the accuracy: (i) improve the accuracy of the integration method, (ii) improve performance using learnings from coordinate-based machine learning processes, and (iii) determine appropriate values for the hyperparameters listed in Section 2.3. Gauss-quadrature integration was employed to improve the accuracy of integration (described in Appendix A). However, a significant increase in accuracy over the trapezoidal rule (used by Nguyen-Thanh et al.[31]) was not observed. Thus, Fourier features mapping and optimization of hyperparameters were simultaneously employed to improve the accuracy. Fourier feature mapping was used to improve calculations during training weights and biases for ANN, as discussed in section 3.1. The optimal values for hyperparameters were obtained using optimization algorithms discussed in Section 3.2.

3.1 Random Fourier Features

Fourier transformation helps convert functions dependent on space or time domain into functions dependent on spatial frequency or temporal frequency. This transformation is widely used in mathematics and digital image processing to simplify calculations. Tancik et al.[44] demonstrated that the standard MLP fails to learn high frequencies, and Fourier features mapping can be used to mitigate this bias. Wang et al.[48] expanded the work of Tancik et al.[44] to PINNs, discussed the limitations of PINNs via the neural tangent kernel theory, and illustrated how PINNs are biased towards learning along the dominant eigenvectors of their limiting NTK. Correspondingly, they have demonstrated that using Fourier feature mappings with PINNs helps modulate the frequency of the NTK eigenvectors.

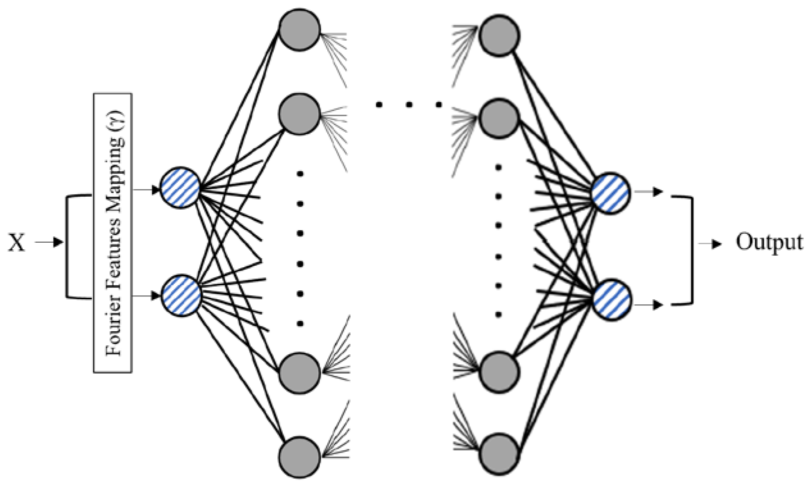


Figure 4: Fourier features mapping in artificial neural network.

Figure 4 shows the implementation of the Fourier feature mapping. The input layer (coordinates of the training points from the physical domain) is passed through a Fourier mapping γ before passing through a multilayer perception. The function γ maps the input coordinates to a higher-dimensional hypersphere using a set of sinusoids:

$$\gamma = [a_1 \cos(2\pi h_l^T X)\, , a_1 \sin(2\pi h_l^T X), \ldots \ldots, a_m \cos(2\pi h_m^T X)\, , a_m \sin(2\pi h_m^T X)] \quad (13)$$

where h_i's are the Fourier basis frequencies used for approximation and a_i represent the corresponding coefficients in the Fourier series. Here, we adopted Gaussian random Fourier features (RFF). Each entry in h is sampled from a normal distribution N (0, σ^2), where σ^2 is the variance. σ is a hyperparameter that needs to be specified for each problem.

3.2 Tuning of hyperparameters

Another area identified to improve the code's accuracy is tuning the hyperparameters. As described earlier, hyperparameters are a set of predefined parameters that control the learning process and affect the model's accuracy. Best combinations of hyperparameters should be used in an ANN to get optimal results. Four primary strategies can be used to obtain the best combinations of hyperparameters. These include manual search, grid search, randomized search, and informed search. Out of the four strategies, manual search involves varying the hyperparameters manually. This approach can reduce search time if the programmer is familiar with the problem and has prior experience with similar models. Otherwise, the manual search can be tedious and time-consuming. In the case of the grid search, the search space defined for hyperparameters is divided into grids. All combinations of hyperparameters are tested to find the best model. This brute force approach can be time-consuming and increases exponentially with the number of variables. The random search is similar to the grid search but reduces the runtime by limiting itself to a predefined number of combinations. The search, however, is random and does not guarantee an optimal set of hyperparameters. Lastly,

informed search utilizes optimization algorithms to obtain the optimal combination for given hyperparameters. The latter is one of the most efficient ways to tune hyperparameters if the programmer does not have prior experience with a similar problem.
A two-loop architecture was employed to develop a systematic approach to tune hyperparameters for DEM, as shown in Figure 5. The inner loop runs DEM to obtain weights and biases for the ANN by training the network over the epochs. The optimal weight and biases obtained after training the ANN provide minimum potential energy of the system for the given ANN architecture. The minimum potential energy of the system is then transferred to the outer loop. The outer loop varies the architecture of ANN by varying hyperparameters (thereby varying the minimum potential energy obtained after training ANN). The objective of the outer loop is to determine values for the hyperparameters for which minimum potential energy of the system is achieved.

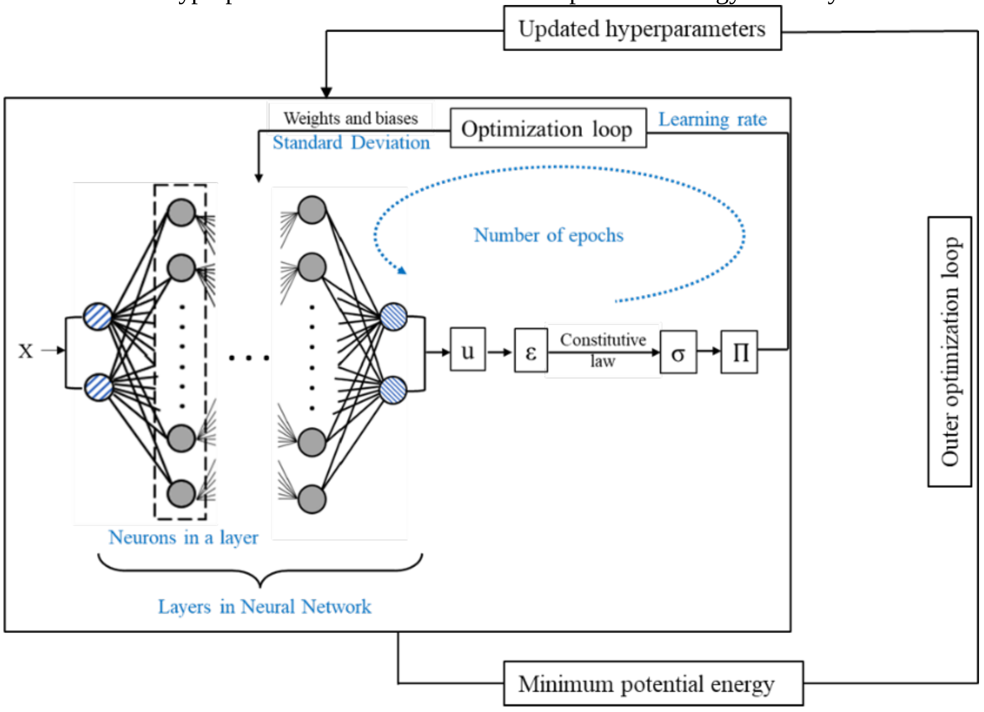


Figure 5: Modified deep energy method.

Several open-source python libraries have been developed to aid in tuning hyperparameters (example: Scikit-learn, Optuna, Keras Tuner, and GPyOpt). For the current study, HyperOpt was chosen as it uses Bayesian optimization algorithms and random search to obtain optimal parameters. Bayesian optimization works well in non-convex optimization problems. Another advantage of HyperOpt is that it can accommodate different types of variables (continuous, ordinal, categorical), different sensitivity profiles (e.g., uniform vs. log scaling), and conditional structure [49]. The underlying algorithm used to obtain optimal hyperparameter is shown in Figure 5.

Table 1: Sensitivity profile and range used for various hyperparameters.

Variable	Sensitivity profile	Range
Learning Rate	loguniform	Exp(0-2)
Neurons	quniform	20-120
Standard Dev (DNN)	uniform	0-1
Standard Dev (FFT)	uniform	0-1
Epochs for L-BFGS optimizer	quniform	40-150
Total number of layers	quniform	3-5
Activation function	choice	Tanh, relu, rrelu, sigmoid

Three main things need to be defined to optimize the hyperparameters: a search domain, an objective function, and the selection of an optimization algorithm. Seven hyperparameters were chosen to be optimized and defined in the search space: total number of layers, number of neurons in hidden layers, standard deviation while choosing weights and biases, activation function, learning rate used for LBFG optimizer, number of iterations for LBFG optimize, and standard deviation used for Fourier features mapping. The search range and sensitivity profile used for these parameters are mentioned in Table 1. The descriptions of the four activation functions mentioned in Table 1 are provided in Appendix C. A profile of *loguniform* returns values drawn from the natural log, such that the natural log of the returned value is uniformly distributed. *Quniform* returns discrete values between upper and lower bounds

specified for the variable. A *uniform* profile returns a value uniformly between the upper and lower bounds, and *choice* is used to sample discrete variables where one of the options specified in the variable space is used for sampling.
The optimization objective was to obtain hyperparameters that minimize the system's potential energy after training DEM. Two optimization algorithms (Tree-structured Parzen Estimator (TPE), adaptive Tree-structured Parzen Estimator (ATPE)) and a random search method were used and compared to determine the best process for optimizing hyperparameters. The details about these algorithms are given in Appendix D. The performance of DEM after obtaining optimal values obtained under different load cases and training algorithms are described in the subsequent section.

4. Tuning hyperparameters for different load cases

The proposed framework, described in Section 3, was used to capture the mechanical deformation for three different load cases specified in Figure 3 and determine optimal hyperparameters for these different BVPs. The optimized hyperparameters for the three load cases obtained using different optimization algorithms are shown in Tables 2-4. All the cases were analyzed using 200x100 training points over the computational domain, as shown in Figure 6. The error in the predicted values from the ones obtained through FEA is calculated using the L_2-error, which is defined as follows:

$$||\, L_2|| = \sqrt{\frac{\sum_{training\ points}\sum_{i=1,2}\left(u_{i(DEM)} - u_{i(FEA)}\right)^2}{number\ of\ training\ points}} \quad (14)$$

Applied boundary conditions and the results for each load case are described in detail in the following subsections. The L_2-error obtained from the three hyperparameter optimization schemes (TPE, ATPE, and random search) were compared with the hyperparameters reported in the literature[31]. The previously used hyperparameters in the literature are referred to as the 'original' hyperparameters in the paper henceforth.

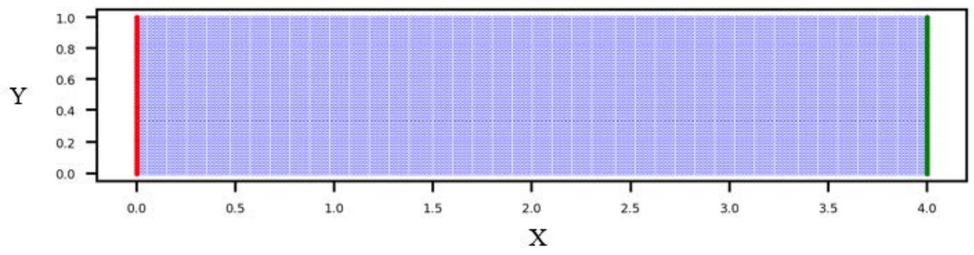


Figure 6: Positions of training points for all load cases studied in Section 4.

4.1 Compressive load case

A uniform compressive load of -50N along the x-axis was distributed uniformly along the right edge of a 4x1 mm^2 rectangular plate. The left edge of the plate was fixed in all degrees of freedom to obtain hyperparameters under compressive loading. The TPE, ATPE, and random search algorithms were employed to predict optimal values for hyperparameters. The results obtained from the three algorithms are shown in Table 2. Figure 7 shows the variation in L_2-error (on log-scale) when the hyperparameters stated in Table 2 are used. As seen in the figure, the L_2-error is reduced by a factor of 10^{-4} when optimal hyperparameters are used.
The displacements along the x- and y-axis for the three sets of hyperparameters (listed in Table 2) and their deviations from FEA are shown in Appendix E. We can notice that hyperparameters obtained from all three algorithms can accurately predict displacements up to an order of 10^{-4} compared to FEA.

Table 2: Optimized hyperparameters for compressive load case.

	Optimal Values			
Variable	**TPE**	**ATPE**	**Random**	**Original**
Learning Rate	5.6322731974	1.7276388866	4.02118946	0.5
Neurons	110	108	82	30
Standard Dev (DNN)	0.00399032891	0.008570979503	0.02152887	0.1
Standard Dev (FFT)	0.76073346011	0.17838495907	0.38436288	-
Epochs for L-BFGS optimizer	88	93	83	80
Total number of layers	4	4	5	4
Activation function	rrelu	rrelu	rrelu	thanh
L_2-error	3.385649e-05	6.396267e-05	6.3503781e-05	0.1034885

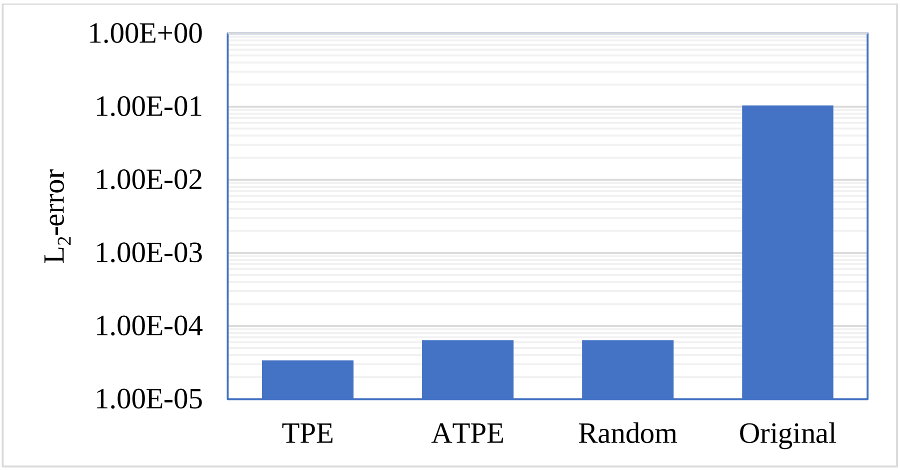


Figure 7: Variation in L_2-error for optimal hyperparameters obtained under compressive loads.

4.2 *Tensile load case*

A uniform tensile load of 50N along the x-axis was distributed uniformly along the right edge of a 4x1 mm^2 rectangular plate. The left edge of the plate was fixed in all degrees of freedom to obtain hyperparameters under tensile load case. The three algorithms (TPE, ATPE, and random search) were used to predict optimal hyperparameters. The optimal values for different hyperparameters are shown in Table 3. The hyperparameters obtained from random search are the same as those obtained under compression load case.

Table 3: Optimized hyperparameters for tensile load case.

	Optimal Values			
Variable	**TPE**	**ATPE**	**Random**	**Original**
Learning Rate	7.018489	1.577630871	4.021189	0.5
Neurons	108	116	82	30
Standard Dev (DNN)	0.000702	0.354804634	0.021529	0.1
Standard Dev (FFT)	0.78729	0.166661609	0.384363	-
Epochs for L-BFGS optimizer	94	75	83	80
Total number of layers	4	5	5	4
Activation function	rrelu	sigmoid	rrelu	thanh
L_2-error	4.518824e-05	0.000394658	5.73980385e-05	0.1265225

Figure 8 shows the variation in L_2-error (on log-scale) when the hyperparameters stated in Table 3 are used. As seen in the figure, the L_2-error is reduced by a factor of 10^{-4} compared to the initial value when TPE and random search algorithms are used and by a factor of 10^{-2} when ATPE is used. The displacements along the x- and y-axis for the three set of hyperparameters (listed in Table 3) are shown in Appendix E. We can notice that hyperparameters obtained from all three algorithms can accurately predict displacements up to an order of 10^{-3} compared to FEA. (The error plots are shown in Appendix E.)

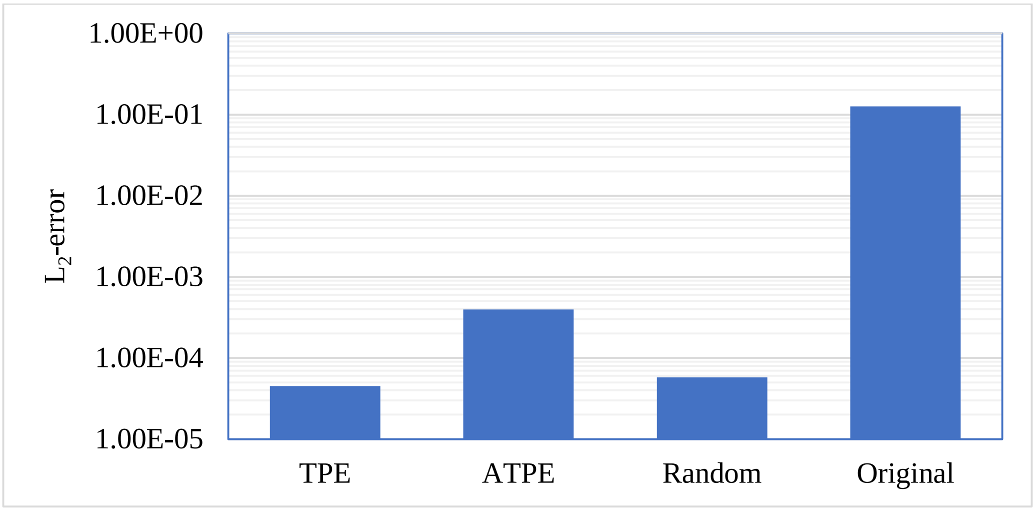


Figure 8: Variation in L_2-Error for optimal hyperparameters obtained under tensile load case.

4.3 *Bending load case*

A uniform bending load of -10N along the y-axis was distributed uniformly along the right edge of a 4x1 mm^2 rectangular plate. The left edge of the plate was fixed in all degrees of freedom. The three algorithms (TPE, ATPE, and random search) were used to predict optimal hyperparameters. The optimal values for different hyperparameters are shown in Table 4.

Figure 9 shows the variation in L_2-Error (on log-scale) for optimal hyperparameters obtained under compressive loads. From Figure 9, we can notice that, in general, the L_2-Error reduces when optimal hyperparameters are used. However, the reduction in the L_2-Error is not as significant as observed in tension and compression load cases. The displacements along the x- and y-axis for the three sets of hyperparameters (listed in Table 4) and corresponding deviations from FEA are shown in Appendix E. We notice that the hyperparameters obtained from all three algorithms predict displacements accurately up to an order of 10^{-3} when compared to FEA. Tables 2-4 show that the deviation of displacement predicted by DEM when compared to FEA is high for bending load case, even for optimal parameters. Also, it can be observed that the displacements in the case of bending are less susceptible to hyperparameters when compared to compression and tension load cases.

Table 4: Optimized hyperparameters for bending load case.

	Optimal Values			
Variables	**TPE**	**ATPE**	**Random**	**Original**
Learning Rate	3.7593948	1.661879	1.136364	0.5
Neurons	120	116	78	30
Standard Dev (DNN)	0.2298224	0.364806	0.008813	0.1
Standard Dev (RNN)	0.1494122	0.396999	0.275733	-
Epochs for L-BFGS optimizer	88	100	95	80
Total number of layers	5	5	3	4
Activation function	tanh	sigmoid	relu	thanh
L_2-error	0.00117923962	0.00190703372	0.00191749373	0.0031548248

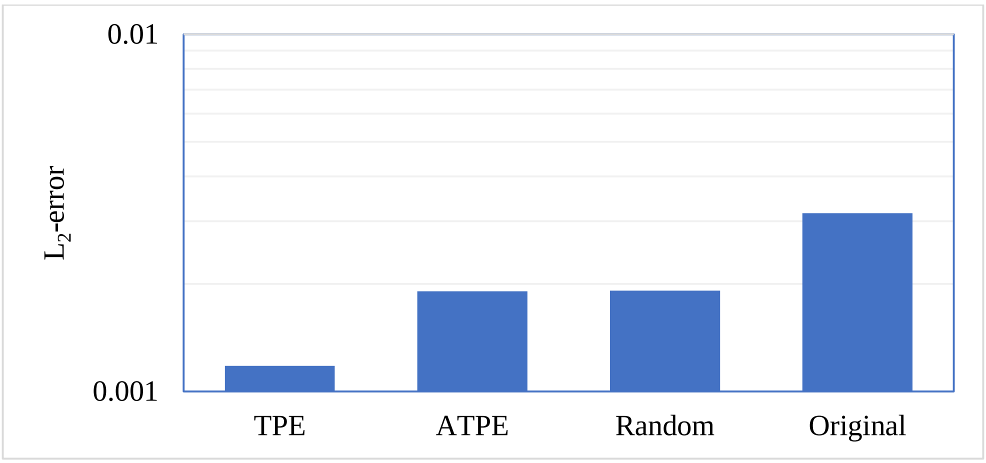


Figure 9: Variation in L_2-error for optimal hyperparameters obtained under bending load case.

5. Transferability of hyperparameters

Though hyperparameters can be optimized for different load cases and geometry of the plate, as shown in Section 4, the optimization process is time-consuming. Thus, in this section, we analyse the transferability of the hyperparameters to different geometries and load cases.

5.1 Transferability of hyperparameters obtained from compression and tension load cases to bending

The optimal hyperparameters obtained from the random search under tension and compression load cases were used to predict displacements under bending loads. Figure 10 shows predicted displacements for all three load cases when hyperparameters obtained from the random search under compression and tension load cases are used to define the architecture of DEM. Even though the parameters were optimized for compressive load case, we notice that it can also predict deformation under uniform tensile and bending loading with L_2-error of 5.73980e-05 and 0.0020859, respectively. Figure 12 compares the L_2-error obtained using these hyperparameters against the optimal ones obtained for bending load case.

5.2 Transferability of hyperparameters obtained from bending load case to compression and tension

The hyperparameters obtained from TPE under bending load case are used to predict displacements under compression and tension loads. The displacements obtained from this hyperparameter set under the three load cases are shown in Figure 11. From solid mechanics, we know that when uniform tensile and compressive loads are applied to a symmetric geometry, the deformation of the geometry perpendicular to the applied load should also be symmetric (along the y-axis in this case). From Figure 11, we can notice that this condition is not met when tensile and compression loads are applied. As a result, the L_2-error is also higher for these loading conditions (Figure 12).

The results presented in sections 5.1 and 5.2 show that the optimal hyperparameters obtained using either tensile or compressive load cases are transferable to bending load case. The values of hyperparameters mentioned in table 2 for the random search algorithm are used to predict deformation in the following sections. The predicted deflections are compared with the results obtained from the finite element analysis undertaken in Abaqus using element type CPS4R. The same mesh density was used between modified DEM and FEA.

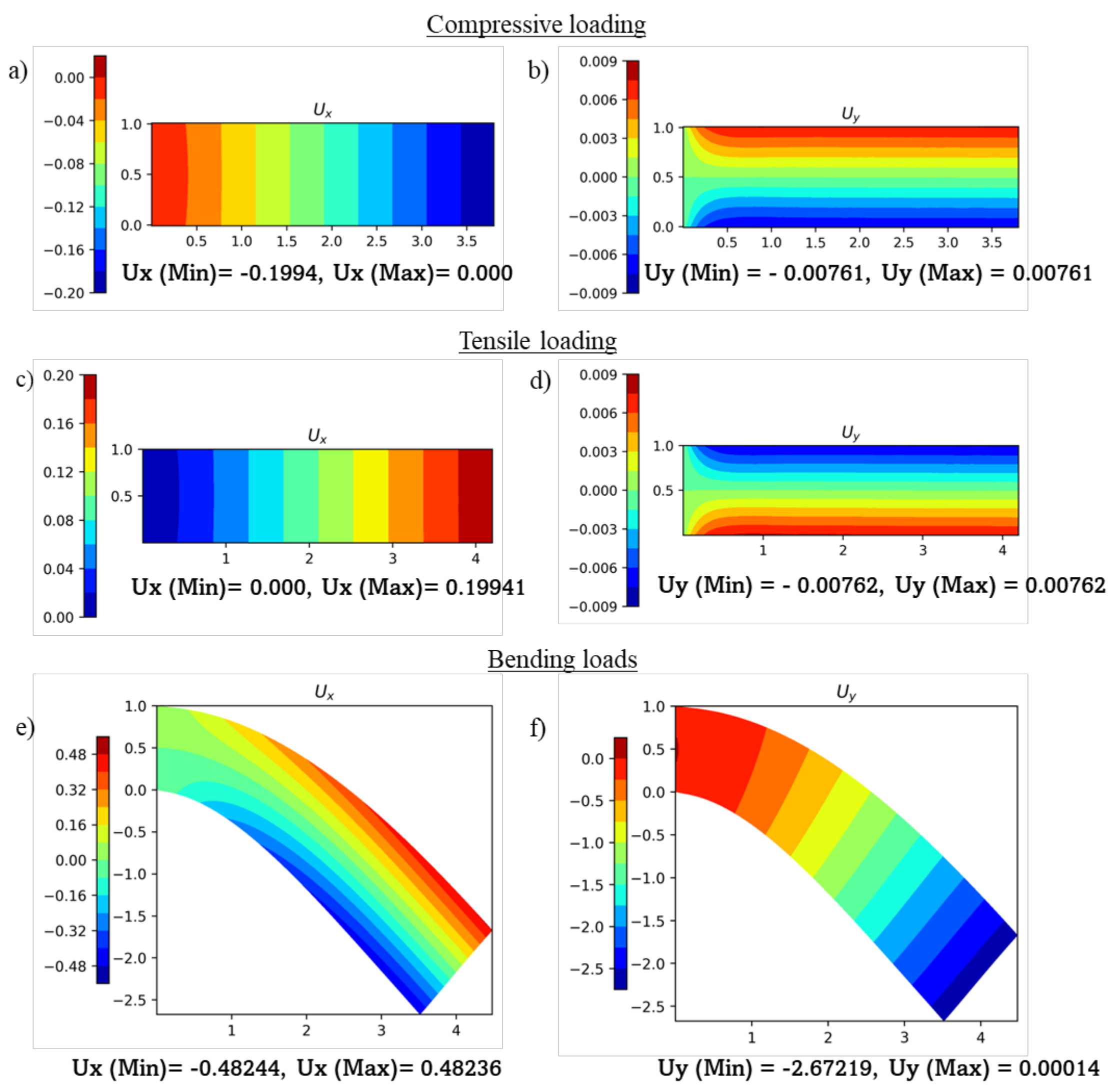


Figure 10: Predicted displacements under compressive (a-b) tensile (c-d) and bending (e-f) loads using optimal hyperparameters obtained from Random search (under tensile and compressive load cases).

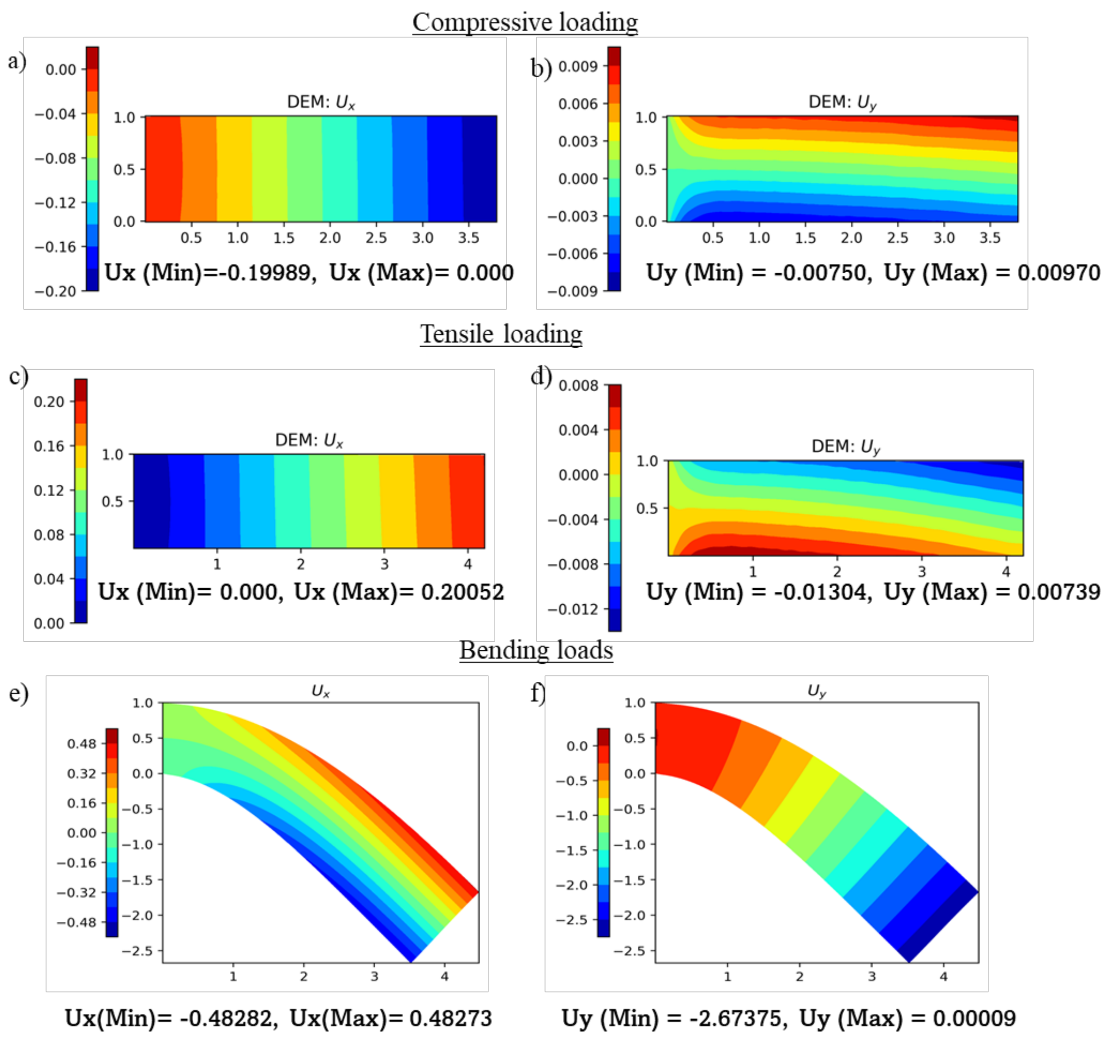


Figure 11: Predicted displacements under bending (a-b), compressive (c-d), and tensile (e-f) loads using optimal hyperparameters obtained from TPE (under bending load case).

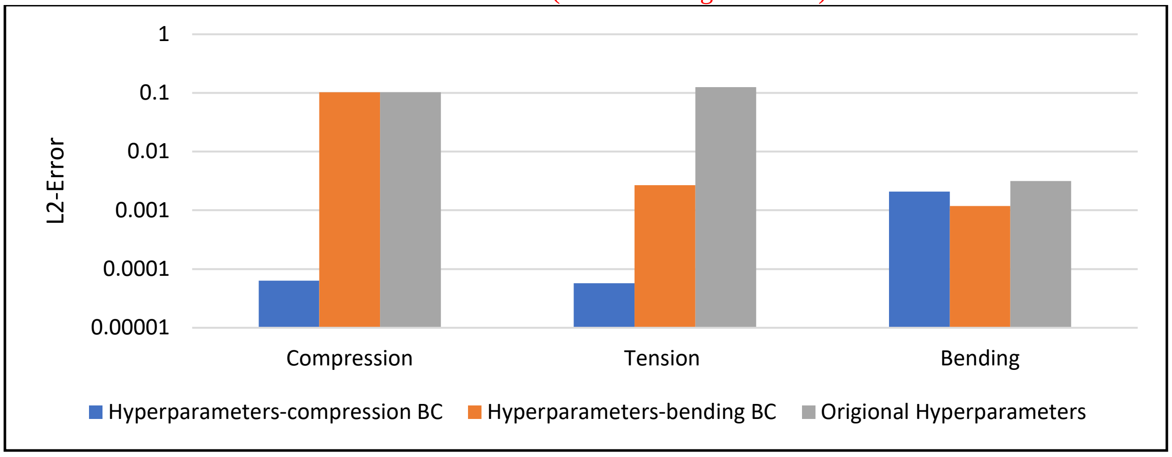


Figure 12: L_2-error using hyperparameters obtained from bending and compression load cases under different loading conditions.

5.3 Transferability across geometry

A plate of 1mm x 1mm cross-section was subjected to uniaxial compression, as shown in Figure 13 a. The training points used for DEM are shown in Figure 13 b.

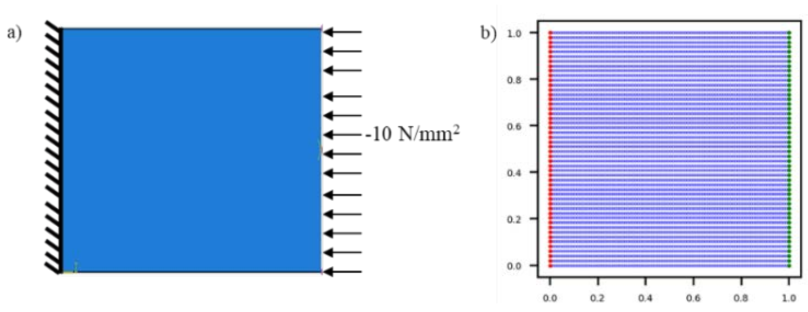


Figure 13: a) 1mmx1mm plate subjected to compressive loads, b) Training points used for DEM.

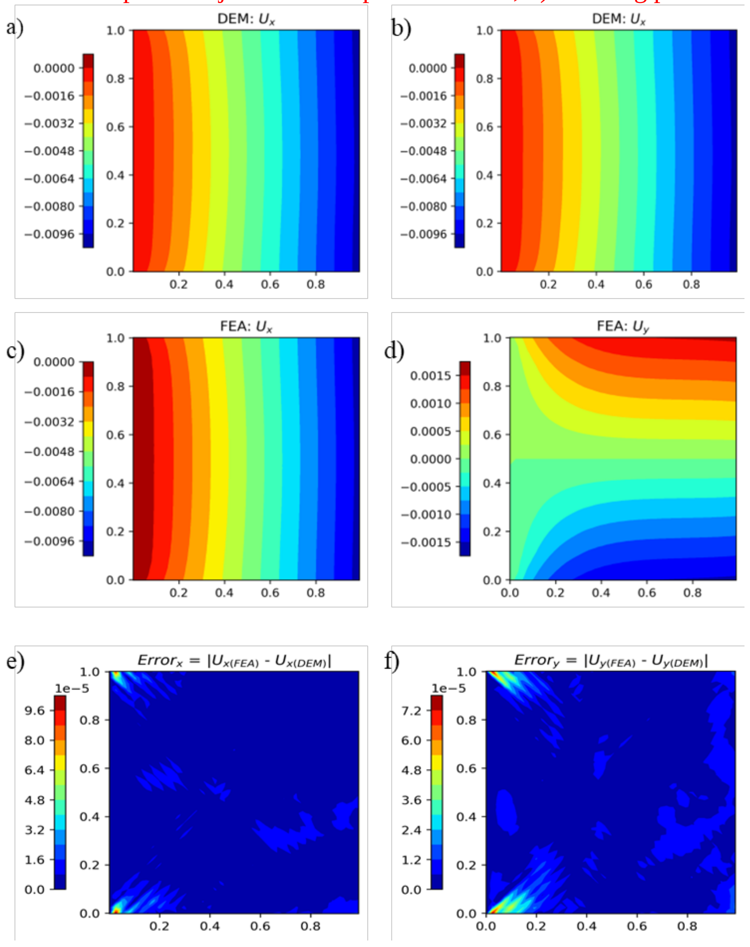


Figure 14: a-b displacements obtained predicted from DEM, c-d displacements obtained from FEA, e-f error in displacements.

From Figure 14, we can conclude that DEM can predict displacement under uniform traction using the optimal hyperparameters parameters obtained from uniform compression for geometry other than the one it was obtained for. We also notice again that the error in displacements are proportional to the magnitude of displacement in each direction.

5.4 *Transferability to localized traction boundary conditions*

In this example, a load is applied locally on the upper right-hand corner of a plate having a 4x1mm^2 cross-section (as shown in Figure 15). The same number of domain and boundary training points are used in Section 4. The displacement obtained from DEM, FEA, and error in prediction compared to FEA are shown in Figure 16.

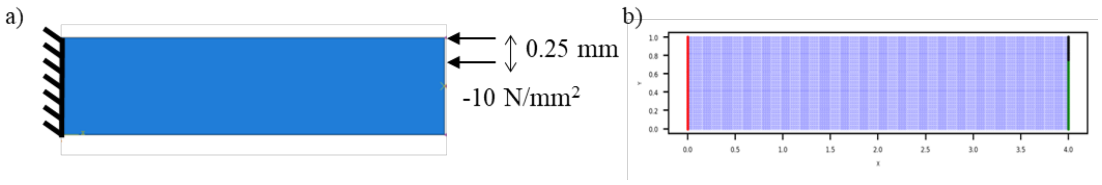


Figure 15: a) Boundary conditions of the plate under localized traction. b) Training points used for DEM.

From Figure 16, we can conclude that DEM can predict displacement under localized traction using the optimal hyperparameters parameters obtained from uniform compression. We also notice again that the error in displacements is proportional to the magnitude of displacement in each direction.

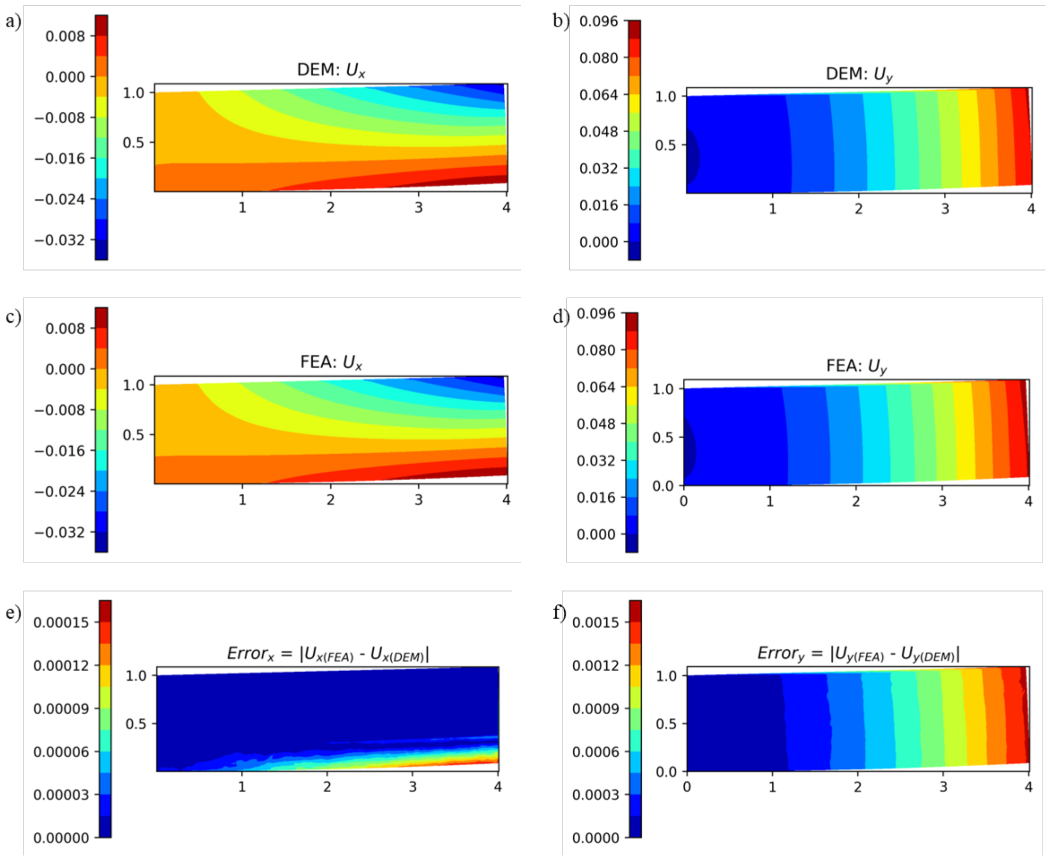


Figure 16: a-b displacements predicted from DEM, c-d displacements obtained from FEA, e-f error in displacements for localized traction.

5.5 *Transferability to plate with a circular hole*

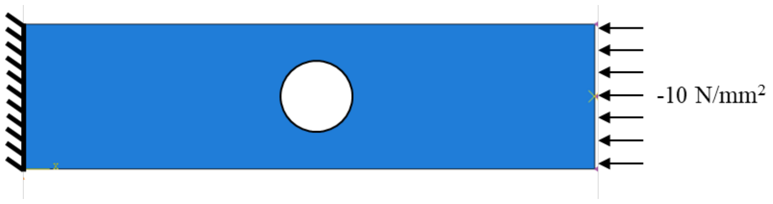


Figure 17: Plate with a hole loaded under uniform compression.

The circular hole, shown in Figure 17, was introduced using passive elements. The passive elements are present during the analysis but do not contribute to internal energy. In modified DEM, the passive elements were introduced by multiplying the corresponding strain energy of the element with their material density. Since passive elements do not contribute to internal energy, the material density corresponding to passive elements was zero. The generalized equation of ellipse (Equation 15) is used to identify passive elements in the case of an elliptical and circular hole.

$$\frac{(x-x_1)^2}{a^2}+\frac{(y-y_1)^2}{b^2}=1 \qquad (15)$$

(x_1, y_1) represents the ellipse's center in the coordinate system. The length of the major and minor axis of the ellipse is represented by *a* and *b*, respectively. A circle is a special case of an ellipse in which *a*=*b*=1.

A uniform compressive load of -10N was applied at the right end of a 4x1mm^2 plate with a circular hole of radius 0.25mm (as shown in Figure 18). The corresponding displacements were obtained from DEM, FEA, and the error between the two is shown in Figure 18. In Figure 18 b, we notice a loss in symmetry in displacement along the y-axis towards the right corner of the plate. However, the model trained using hyperparameters obtained from uniform compressive loads can predict displacements, up to an order of 10^{-3}, when compared with FEA.

6 Conclusions

This paper proposes a two-loop architecture to obtain the best architecture and hyperparameters for DEM. The inner loop predicts displacements and potential energy using DEM based on hyperparameters chosen by the outer loop. The outer loop receives the system's potential energy from the inner loop, which is dependent on architecture of ANN defined by hyperparameters. The optimization algorithm in the outer loop then searches the non-convex optimization space to identify hyperparameters for which minimum potential energy can be achieved. A Fourier features mapping is employed in the inner loop (DEM) to improve accuracy and simplify calculations. Seven hyperparameters were chosen for the study. It was found that the values of hyperparameters identified using this approach and the implementation of Fourier feature mapping produce displacements similar to those observed through FEA (with an order of accuracy of O(h^{-3})). Additionally, based on the examples presented in the paper, we can conclude the following:

- The displacements obtained for tension and compression load cases are more sensitive to hyperparameters than displacements obtained for bending loads.
- With optimal hyperparameters, the order of accuracy for tension and compression load cases is lower than for bending loads.
- The optimal hyperparameters chosen through compression and tension load cases can predict displacement for various geometry and loading conditions. As a result, the optimal hyperparameters can be searched for a single condition (depending on the desired accuracy).
- The error in predicted displacements is proportional to the magnitude of displacement.
- In general, Randomized Leaky Rectified Linear Unit (rrelu) was found to be the best choice for activation function.
- rrelu, along with five layers and ~80 neurons, produce the least potential energy under all three loading conditions.

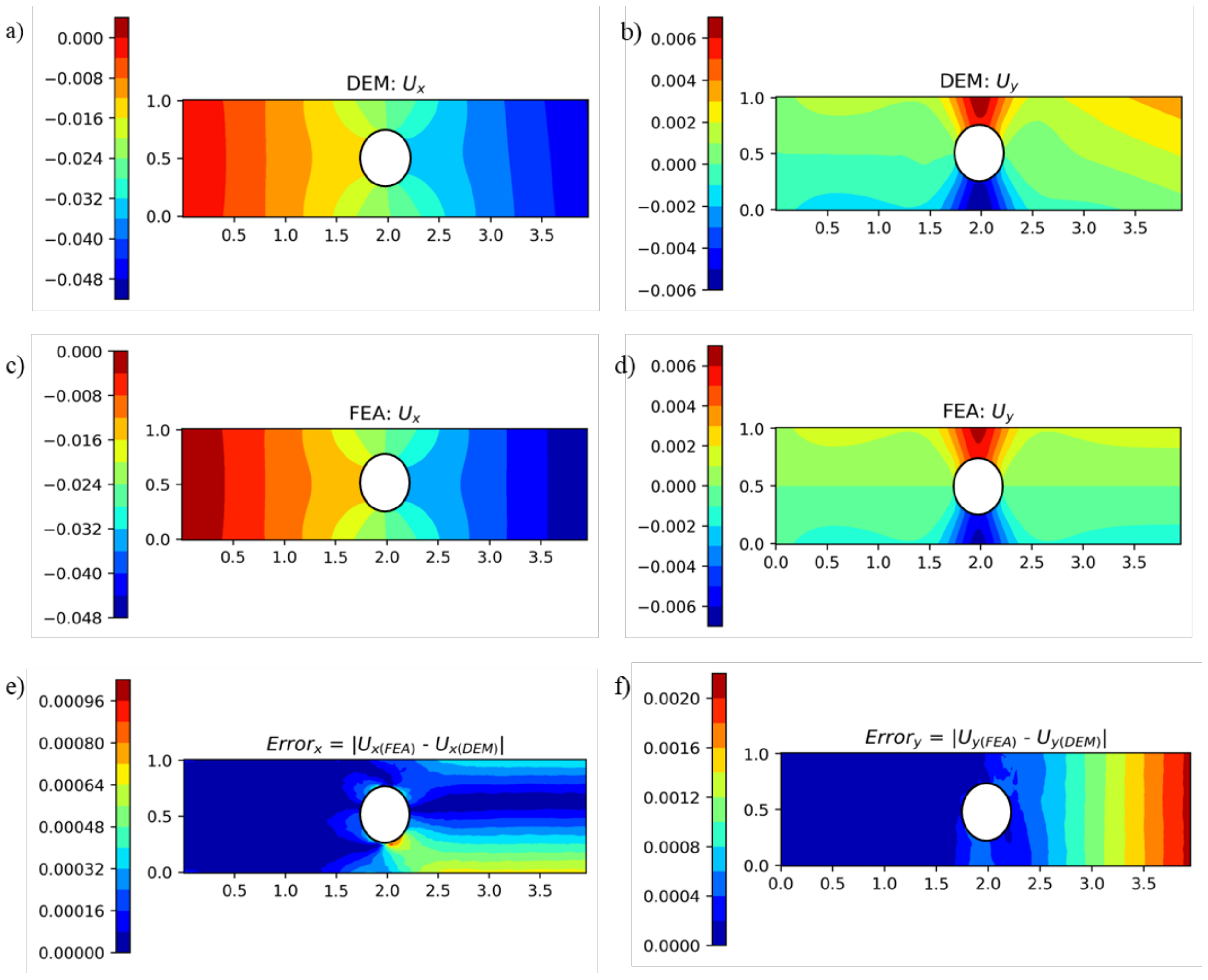


Figure 19: a-b displacements predicted from DEM, c-d displacements obtained from FEA, e-f error in displacements for a plate with a hole subjected to uniform traction.

Acknowledgments:
The authors would like to thank the National Center for Supercomputing Applications (NCSA) Industry Program and the Centre for Artificial Intelligence Innovation for funding this research.

Data availability:
The data supporting this study's findings are available from the corresponding author upon reasonable request.

Code availability:
The code for replication of our experiments will be available upon acceptance.

References:

1. Belytschko T, Rabczuk T, Huerta A, Fernández-Méndez S. Meshfree Methods. In: *Encyclopedia of Computational Mechanics*. John Wiley & Sons, Ltd; 2004. doi:10.1002/0470091355.ecm005
2. Belytschko T, Krysl P, Krongauz Y. A three-dimensional explicit element-free galerkin method. *Int J Numer Methods Fluids*. 1997;24(12):1253-1270. doi:10.1002/(SICI)1097-0363(199706)24:12<1253::AID-FLD558>3.0.CO;2-Z
3. Hughes TJR. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publications; 2012.
4. Hughes TJR, Sangalli G, Tani M. Isogeometric Analysis: Mathematical and Implementational Aspects, with Applications. In: ; 2018:237-315. doi:10.1007/978-3-319-94911-6_4
5. Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput Methods Appl Mech Eng*. 2005;194(39-41):4135-4195. doi:10.1016/j.cma.2004.10.008
6. Kim Y, Kim Y, Yang C, Park K, Gu GX, Ryu S. Deep learning framework for material design space exploration using active transfer learning and data augmentation. *npj Comput Mater*. 2021;7(1):140. doi:10.1038/s41524-021-00609-2
7. Rong Q, Wei H, Huang X, Bao H. Predicting the effective thermal conductivity of composites from cross sections images using deep learning methods. *Compos Sci Technol*. 2019;184:107861. doi:10.1016/j.compscitech.2019.107861
8. Liu WK, Karniadakis G, Tang S, Yvonnet J. A computational mechanics special issue on: data-driven modeling and simulation—theory, methods, and applications. *Comput Mech*. 2019;64(2):275-277. doi:10.1007/s00466-019-01741-z
9. Mozaffar M, Bostanabad R, Chen W, Ehmann K, Cao J, Bessa MA. Deep learning predicts path-dependent plasticity. *Proc*

Natl Acad Sci. 2019;116(52):26414-26420. doi:10.1073/pnas.1911815116

10. Koric S, Abueidda DW. Deep learning sequence methods in multiphysics modeling of steel solidification. *Metals (Basel)*. 2021;11(3):494. doi:10.3390/met11030494
11. Fatehi E, Yazdani Sarvestani H, Ashrafi B, Akbarzadeh AH. Accelerated design of architectured ceramics with tunable thermal resistance via a hybrid machine learning and finite element approach. *Mater Des*. 2021;210:110056. doi:10.1016/j.matdes.2021.110056
12. Spear AD, Kalidindi SR, Meredig B, Kontsos A, le Graverend J-B. Data-driven materials investigations: the next frontier in understanding and predicting fatigue behavior. *JOM*. 2018;70(7):1143-1146. doi:10.1007/s11837-018-2894-0
13. Gu GX, Chen C-T, Richmond DJ, Buehler MJ. Bioinspired hierarchical composite design using machine learning: simulation, additive manufacturing, and experiment. *Mater Horizons*. 2018;5(5):939-945. doi:10.1039/C8MH00653A
14. Linka K, Hillgärtner M, Abdolazizi KP, Aydin RC, Itskov M, Cyron CJ. Constitutive artificial neural networks: A fast and general approach to predictive data-driven constitutive modeling by deep learning. *J Comput Phys*. 2021;429:110010. doi:10.1016/j.jcp.2020.110010
15. Abueidda DW, Koric S, Sobh NA, Sehitoglu H. Deep learning for plasticity and thermo-viscoplasticity. *Int J Plast*. 2021;136:102852. doi:10.1016/j.ijplas.2020.102852
16. Qiu H, Yang H, Elkhodary K l., Tang S, Guo X, Huang J. A data-driven approach for modeling tension–compression asymmetric material behavior: numerical simulation and experiment. *Comput Mech*. 2022;69(1):299-313. doi:10.1007/s00466-021-02094-2
17. 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. doi:10.1016/j.jcp.2018.10.045
18. Abueidda DW, Lu Q, Koric S. Meshless physics-informed deep learning method for three-dimensional solid mechanics. *Int J Numer Methods Eng*. 2021;122(23):7182-7201. doi:10.1002/nme.6828
19. Haghighat E, Raissi M, Moure A, Gomez H, Juanes R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comput Methods Appl Mech Eng*. 2021;379:113741. doi:10.1016/j.cma.2021.113741
20. Cai S, Wang Z, Wang S, Perdikaris P, Karniadakis GE. Physics-informed neural networks for heat transfer problems. *J Heat Transfer*. 2021;143(6). doi:10.1115/1.4050542
21. Hamdia KM, Zhuang X, Rabczuk T. An efficient optimization approach for designing machine learning models based on genetic algorithm. *Neural Comput Appl*. 2021;33(6):1923-1933. doi:10.1007/s00521-020-05035-x
22. Henkes A, Wessels H, Mahnken R. Physics informed neural networks for continuum micromechanics. *Comput Methods Appl Mech Eng*. 2022;393:114790. doi:10.1016/j.cma.2022.114790
23. Amini Niaki S, Haghighat E, Campbell T, Poursartip A, Vaziri R. Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture. *Comput Methods Appl Mech Eng*. 2021;384:113959. doi:10.1016/j.cma.2021.113959
24. Rao C, Sun H, Liu Y. Physics-informed deep learning for computational elastodynamics without labeled data. *J Eng Mech*. 2021;147(8):04021043. doi:10.1061/(ASCE)EM.1943-7889.0001947
25. Himanen L, Geurts A, Foster AS, Rinke P. Data-driven materials science: status, challenges, and perspectives. *Adv Sci*. 2019;6(21):1900808. doi:10.1002/advs.201900808
26. Flaschel M, Kumar S, De Lorenzis L. Unsupervised discovery of interpretable hyperelastic constitutive laws. *Comput Methods Appl Mech Eng*. 2021;381:113852. doi:10.1016/j.cma.2021.113852
27. Yang H, Xiang Q, Tang S, Guo X. Learning material law from displacement fields by artificial neural network. *Theor Appl Mech Lett*. 2020;10(3):202-206. doi:10.1016/j.taml.2020.01.038
28. Chen G. Recurrent neural networks (RNNs) learn the constitutive law of viscoelasticity. *Comput Mech*. 2021;67(3):1009-1019. doi:10.1007/s00466-021-01981-y
29. Lin J, Zhou S, Guo H. A deep collocation method for heat transfer in porous media: Verification from the finite element method. *J Energy Storage*. 2020;28:101280. doi:10.1016/j.est.2020.101280
30. Samaniego E, Anitescu C, Goswami S, 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. doi:10.1016/j.cma.2019.112790
31. Nguyen-Thanh VM, Zhuang X, Rabczuk T. A deep energy method for finite deformation hyperelasticity. *Eur J Mech - A/Solids*. 2020;80:103874. doi:10.1016/j.euromechsol.2019.103874
32. Nguyen-Thanh VM, Anitescu C, Alajlan N, Rabczuk T, Zhuang X. Parametric deep energy approach for elasticity accounting for strain gradient effects. *Comput Methods Appl Mech Eng*. 2021;386:114096. doi:10.1016/j.cma.2021.114096
33. Wang S, Yu X, Perdikaris P. When and why PINNs fail to train: A neural tangent kernel perspective. *J Comput Phys*. 2022;449:110768. doi:10.1016/j.jcp.2021.110768
34. Wang S, Teng Y, Perdikaris P. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM J Sci Comput*. 2021;43(5):A3055-A3081. doi:10.1137/20M1318043
35. Haghighat E, Amini D, Juanes R. Physics-informed neural network simulation of multiphase poroelasticity using stress-split sequential training. *Comput Methods Appl Mech Eng*. 2022;397:115141. doi:10.1016/j.cma.2022.115141

36. E W, Yu B. The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. *Commun Math Stat*. 2018;6(1):1-12. doi:10.1007/s40304-018-0127-z
37. Abueidda DW, Koric S, Al-Rub RA, Parrott CM, James KA, Sobh NA. A deep learning energy method for hyperelasticity and viscoelasticity. *Eur J Mech - A/Solids*. 2022;95:104639. doi:10.1016/j.euromechsol.2022.104639
38. Fuhg JN, Bouklas N. The mixed Deep Energy Method for resolving concentration features in finite strain hyperelasticity. *J Comput Phys*. 2022;451:110839. doi:10.1016/j.jcp.2021.110839
39. Abueidda DW, Guleryuz E, Sobh NA. Enhanced physics-informed neural networks for hyperelasticity a preprint. *arXiv preprint arXiv:2205.14148*; 2022.
40. Bergstra J, Bardenet R, Bengio Y, Kegl B. Algorithms for hyper-parameter optimization. In: *Proceedings Neural Information and Processing Systems*. ; 2011:2546-2554.
41. Yu T, Zhu H. Hyper-parameter optimization: A review of algorithms and applications. *arXiv Prepr arXiv200305689*. Published online 2020.
42. Luo G. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Netw Model Anal Heal Informatics Bioinforma*. 2016;5(1):18. doi:10.1007/s13721-016-0125-6
43. Shahriari B, Swersky K, Wang Z, Adams RP, de Freitas N. Taking the human out of the loop: a review of bayesian optimization. *Proc IEEE*. 2016;104(1):148-175. doi:10.1109/JPROC.2015.2494218
44. Tancik M, Srinivasan PP, Mildenhall B, et al. Fourier features let networks learn high frequency functions in low dimensional domains. *Adv Neural Inf Process Syst*. 2020;33:7537-7547.
45. Rao SS. *The Finite Element Method in Engineering*. Elsevier Science; 2011.
46. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Networks*. 1989;2(5):359-366. doi:10.1016/0893-6080(89)90020-8
47. Funahashi K-I. On the approximate realization of continuous mappings by neural networks. *Neural Networks*. 1989;2(3):183-192. doi:10.1016/0893-6080(89)90003-8
48. 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. doi:10.1016/j.cma.2021.113938
49. Komer B, Bergstra J, Eliasmith C. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In: ; 2014:32-37. doi:10.25080/Majora-14bd3278-006
50. Smith M. *ABAQUS/Standard User's Manual, Version 6.9*. Dassault Systèmes Simulia Corp; 2009.
51. Bergstra J, Bardenet R, Bengio Y, Kégl B. Algorithms for Hyper-Parameter Optimization. In: Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger KQ, eds. *Advances in Neural Information Processing Systems*. Vol 24. Curran Associates, Inc.; 2011. https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
52. Wen L, Ye X, Gao L. A new automatic machine learning based hyperparameter optimization for workpiece quality prediction. *Meas Control*. 2020;53(7-8):1088-1098. doi:10.1177/0020294020932347
53. Jones DR. *A Taxonomy of Global Optimization Methods Based on Response Surfaces*. Vol 21.; 2001.

doi:10.20944/preprints202206.0414.v1

Appendix A: Gauss-quadrature integration scheme

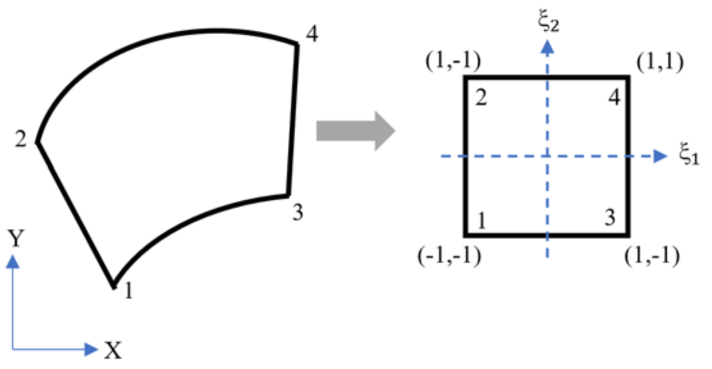


Figure A1: Isoparametric mapping

Gauss-quadrature integration is used in numerical analysis to evaluate the definite integral of a function. An n-point Gauss quadrature can accurately evaluate a definite integral for a degree 2n-1 or less polynomial by finding the value of the polynomial at the suitable choice of nodes x_i. The integration value is found by summing the value of the polynomial at x_i for all n nodes after multiplying it with their respective weights (Equation 13). Such approach is widely used in FEA. The current study used the formulation for isoparametric square elements inspired by FEA to find the system's internal energy (Equation 14).

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

$$U = \int_B \psi(X)dV_x = \sum_{\mathrm{e}=1}^{N} \psi^e(X)dV_x = \sum_{\Omega_\xi} \psi(\xi)|J|dV_\xi \qquad \text{A1}$$

where J is the Jacobian defined by:

$$J = \begin{bmatrix} \frac{\partial X_1}{\partial \xi_1} & \frac{\partial X_2}{\partial \xi_1} \\ \frac{\partial X_1}{\partial \xi_2} & \frac{\partial X_2}{\partial \xi_2} \end{bmatrix} \qquad \text{A2} \quad \text{A3}$$

Isoparametric elements use natural (or intrinsic) coordinate systems to formulate equations defined by the element's geometry and not the orientation or placement of the element in the global coordinate system using shape functions. As shown in Figure 7, transformation mapping using the Jacobian matrix is used to develop relationships between the natural coordinates and the coordinate system. Shape functions for an isoparametric first order square function are defined as:

$$N_1 = \frac{(1-\xi_1)(1-\xi_2)}{4}$$
$$N_2 = \frac{(1-\xi_1)(1+\xi_2)}{4}$$
$$N_3 = \frac{(1+\xi_1)(1-\xi_2)}{4}$$
$$N_4 = \frac{(1+\xi_1)(1+\xi_2)}{4} \qquad \text{A4}$$

The necessary derivatives with respect to the physical coordinate system can be obtained using the chain rule:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = J^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi_1} \\ \frac{\partial N_i}{\partial \xi_2} \end{Bmatrix} \qquad \text{A5}$$

Using the above relationship, the strains in the system can be calculated by

$$\epsilon = [B]\{u\}$$

where,

A6

$$[B] = \begin{bmatrix} \frac{\partial N_1}{\partial \xi_1} & 0 & \frac{\partial N_2}{\partial \xi_1} & 0 & \frac{\partial N_3}{\partial \xi_1} & 0 & \frac{\partial N_4}{\partial \xi_1} & 0 \\ 0 & \frac{\partial N_1}{\partial \xi_2} & 0 & \frac{\partial N_2}{\partial \xi_2} & 0 & \frac{\partial N_3}{\partial \xi_2} & 0 & \frac{\partial N_4}{\partial \xi_2} \\ \frac{\partial N_1}{\partial \xi_2} & \frac{\partial N_1}{\partial \xi_1} & \frac{\partial N_2}{\partial \xi_2} & \frac{\partial N_2}{\partial \xi_1} & \frac{\partial N_3}{\partial \xi_2} & \frac{\partial N_3}{\partial \xi_1} & \frac{\partial N_4}{\partial \xi_2} & \frac{\partial N_4}{\partial \xi_1} \end{bmatrix}$$

The corresponding stresses and strain energy density function are calculated using Equations 8 and 5. The total internal energy is the submission of the strain energy density for all elements.

$$U = \sum_{\Omega_\xi} \psi(\xi)|J|dV_\xi = \sum_{\Omega_\xi} \sum_{i=1}^{4} \sum_{j=1}^{4} W_i W_j \psi(\xi_1, \xi_2)|J|dV_\xi \tag{A7}$$

Appendix B: Deflections obtained before including Fourier feature mapping and optimizing hyperparameters

Figures B1 and B2 show the deflection of the plate and strains along the x-axis and y-axis for the three loading conditions (shown in Figure 3). From Figure B1, we can notice that the displacement in the y-axis is not symmetric around the middle of the plate under compression and tension, which should be the case under symmetric loading. The discrepancy in displacements results in an inaccurate prediction of stains in Figure B2. Error in displacements from FEA is shown in Figure B3. Figure B4 shows the results obtained from Abaqus[50] under three different loading conditions. We can notice that even though the code accurately predicts deflections in bending, it has a high error in prediction in the case of compression and tension.

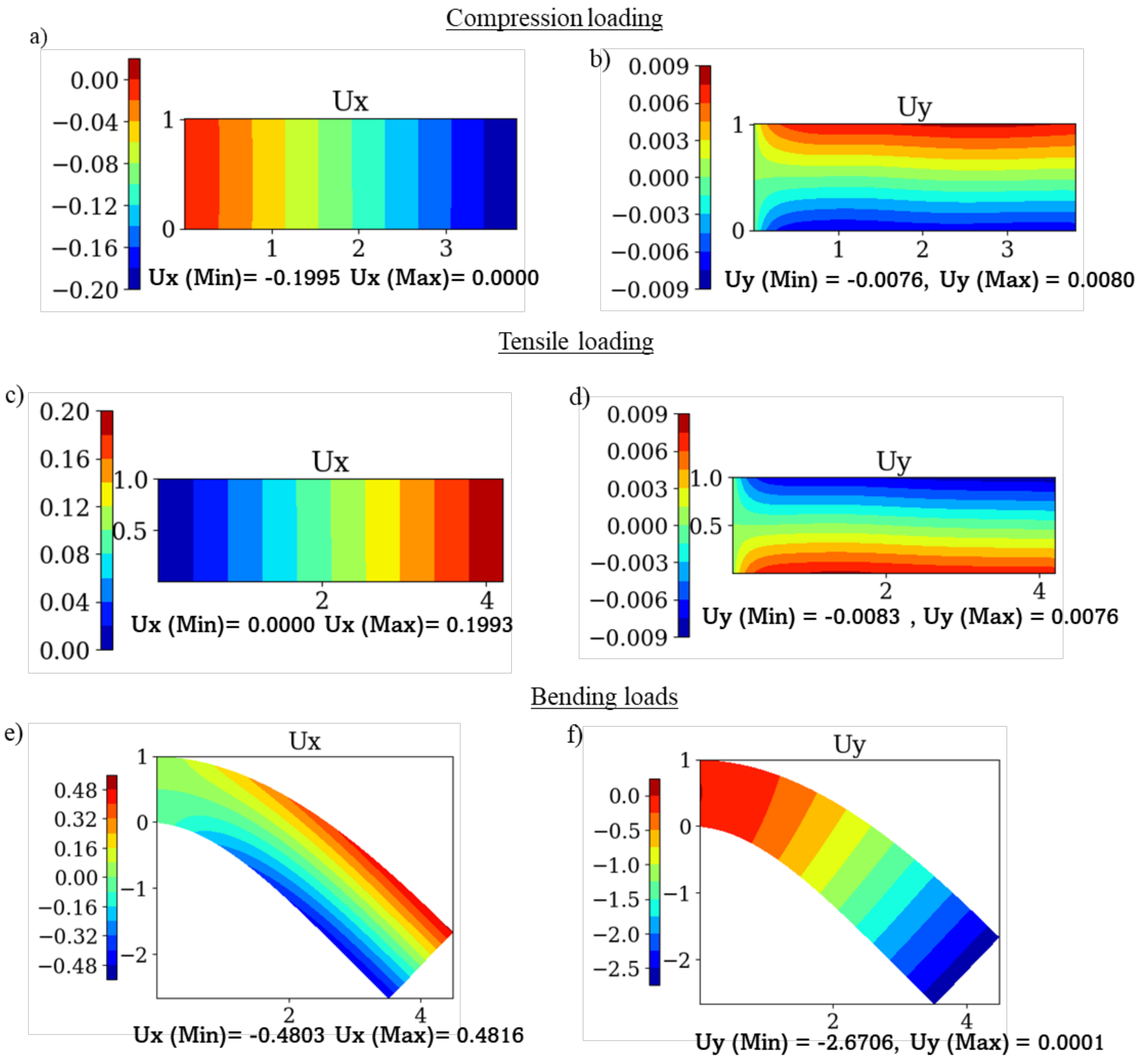


Figure B1: Displacement obtained in x and y directions from DEM without modifications

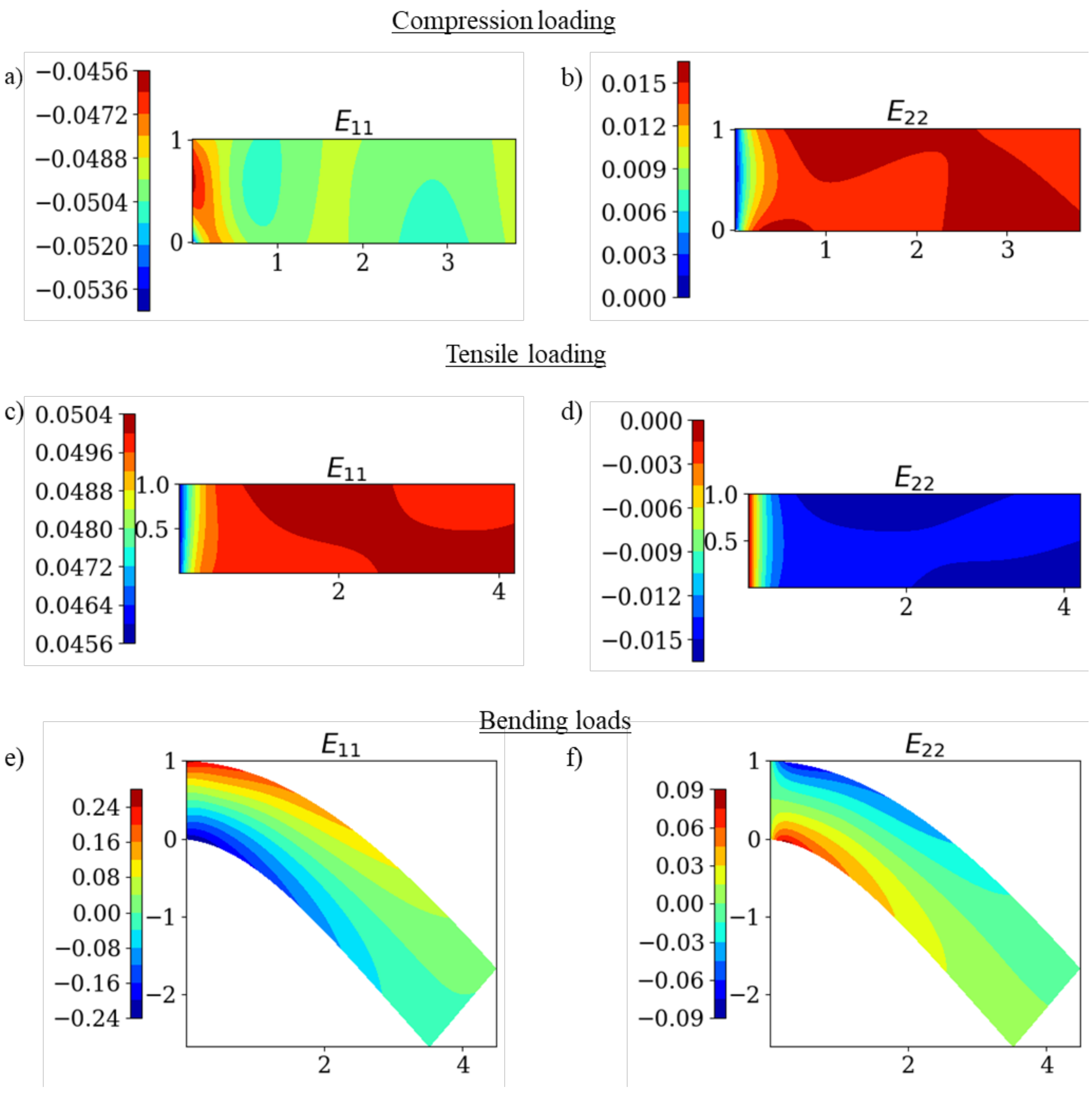


Figure B2: Strains (ε_{xx} and ε_{yy}) obtained from DEM without modifications for compressive loads (a-b), tension loads (c-d), and bending loads (e-f)

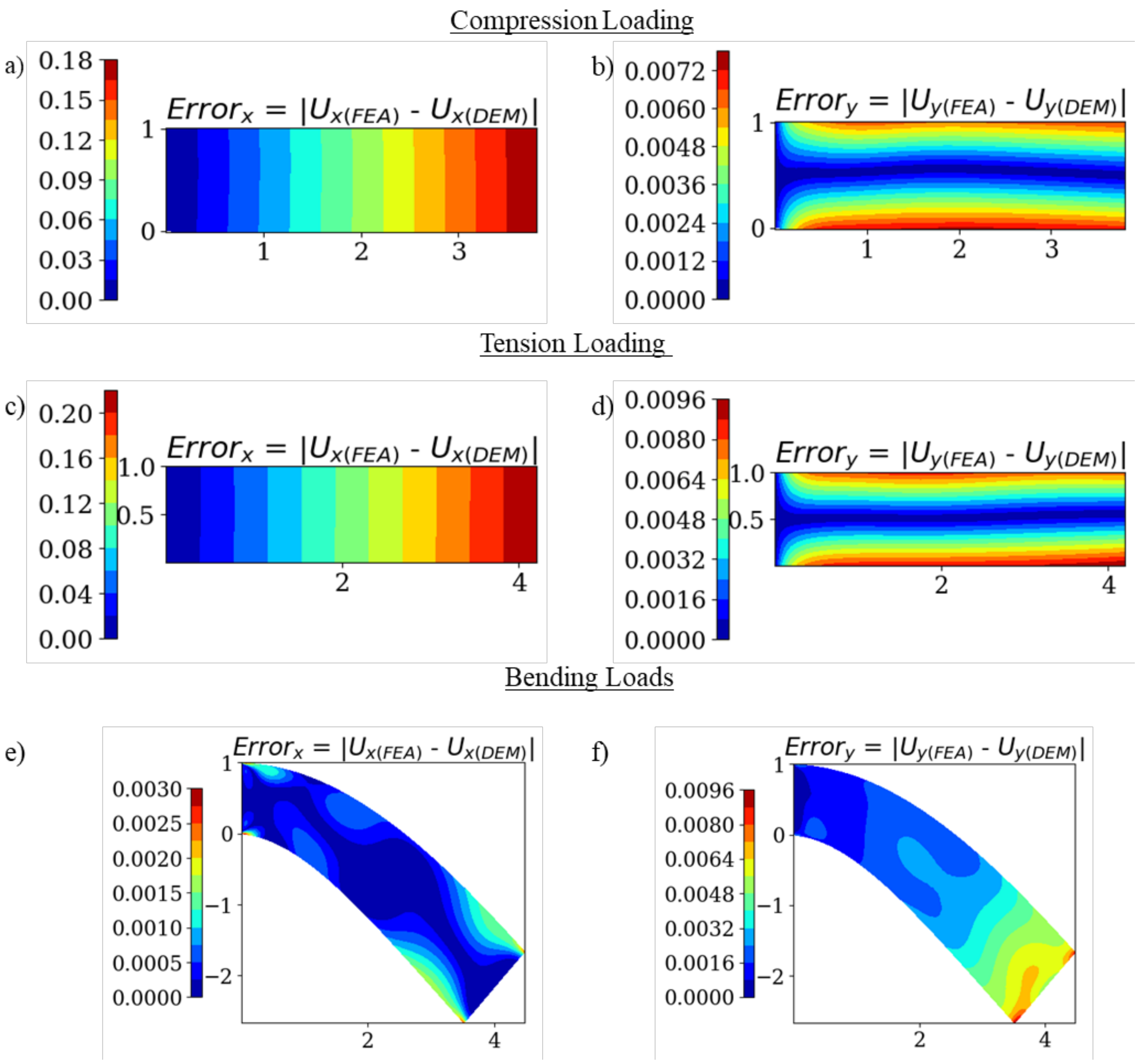


Figure B3: Error in displacements in x and y directions when compared to FEA

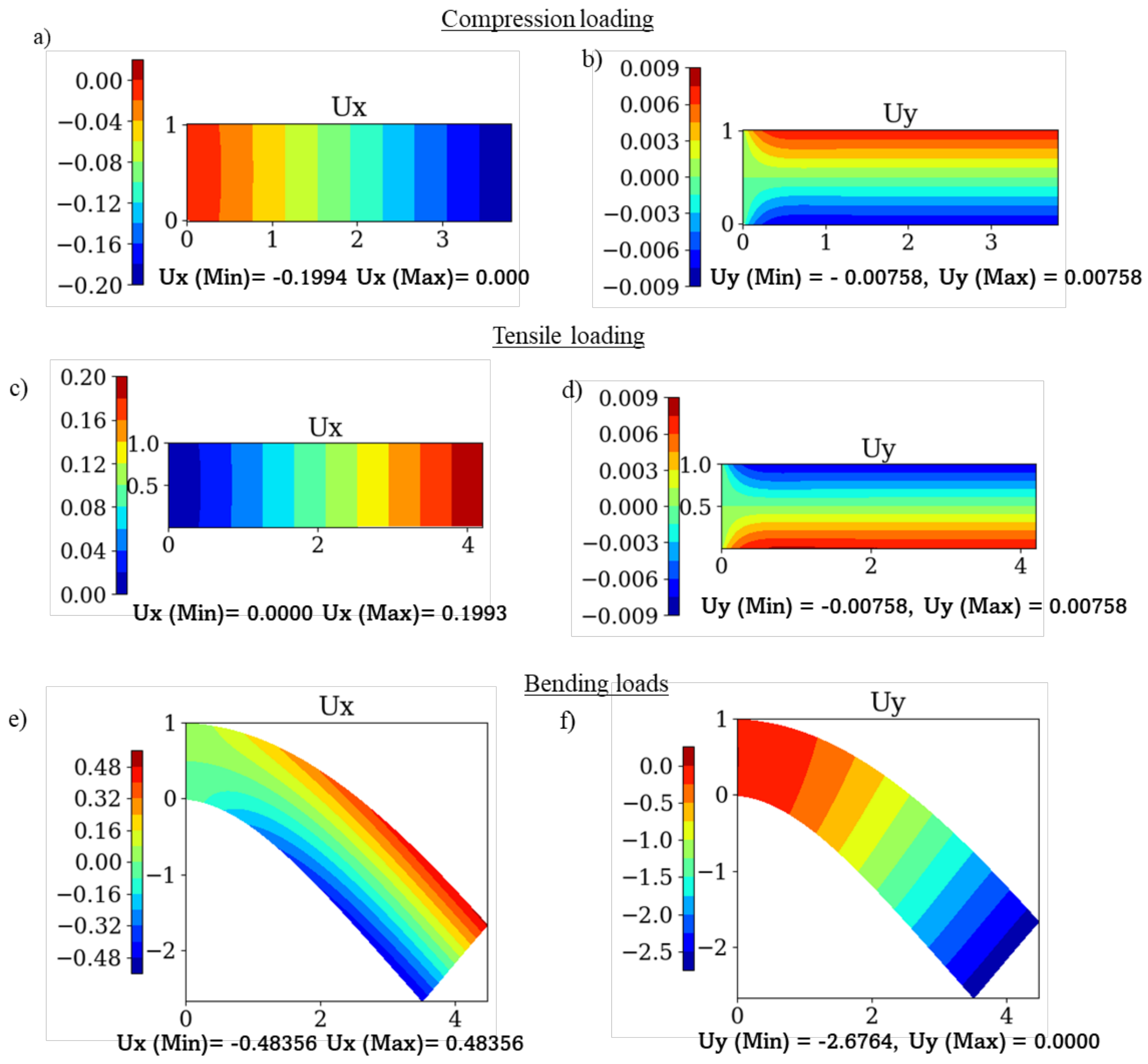


Figure B4: Displacements obtained from FEA

Appendix C: Activation functions

Tanh:

$$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Rectified linear activation function (ReLU):

$$y = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Randomized rectified linear activation function (RReLU):

$$y = \begin{cases} x & , x \geq 0 \\ a * x & , x < 0 \end{cases}$$

where, a is a random number sampled from a uniform distribution

Sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Appendix D: Hyperparameters optimization algorithms

Two algorithms, tree pazen estimator (TPE) and adaptive tree pazen estimator (ATPE) are used to obtain optimal hyperparameters. Both algorithms are based on the Bayesian optimization technique. Bayesian optimization used the Bayes theorem to direct the search for optimal parameters to an objective function's global maximum or minimum value[43]. A probability-based surrogate model of the objective function is constructed from the posterior probability distribution to direct the search. The surrogate model is used to select possible candidates to be sampled. The surrogate model is then updated based on the sampled points. The process continues till the termination criterion for the algorithm is met. Bayesian optimization is beneficial when evaluations of the objective function are costly and the search space is non-convex [43].

The main algorithm for tree pazen estimator (TPE) and adaptive tree pazen estimator (ATPE) is shown below. The readers are directed to publications by Bergstra et al.[51]and Wen et al.[52] for a detailed description of TPE and ATPE, respectively.

D.1 Tree pazen estimator

TPE was proposed by Bergstra et al. [51]. The estimator models both p(x|y) and p(y) to reduce computations. It transforms the prior distribution of the search space into truncated Gaussian distribution and modifies the posterior distribution based on observations. It then sorts the target value (y) and divides it into two, as shown in Equation D1.

$$p(x|y) = \begin{cases} l(x) & , y < y^* \\ g(x) & , y \geq y^* \end{cases} \tag{D1}$$

Where y^* is the boundary used to segregate the target value, $l(x)$ is the density formed by observations that are less than $y^*$and g(x) is the density formed by the remaining observations. The TPE algorithm chooses y^* to be some quantile γ of the observed y values so that $p(y < y^*) = \gamma$. The next promising point is chosen based on the criterion of Expected Improvement (EI)[53].

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y)p(y|x)\,dy = \int_{-\infty}^{y^*} (y^* - y)\frac{p(y|x)p(y)}{p(x)}dy \tag{D2}$$

The main algorithm is shown in Figure D.1.

```
Algorithm 1: Tree pazen estimator
Input: Configuration space, Objective function
Output: Sample and evaluate some points, construct observation set D; $(x_i, y_i) \in D$
Start:
        1. Fit D by probabilistic surrogate model
        2. Determine the next point to be evaluated based on EI ($x_{i+1}$)
        3. Evaluate the chosen point ($y_{i+1}$)
        4. Update the observation set D
        5. Increment i
if i<n: go to step 1
Else: Output the best hyperparameters and function values
End
```

Figure D.1 Algorithm for TPE

D.2 Adaptive tree pazen estimator

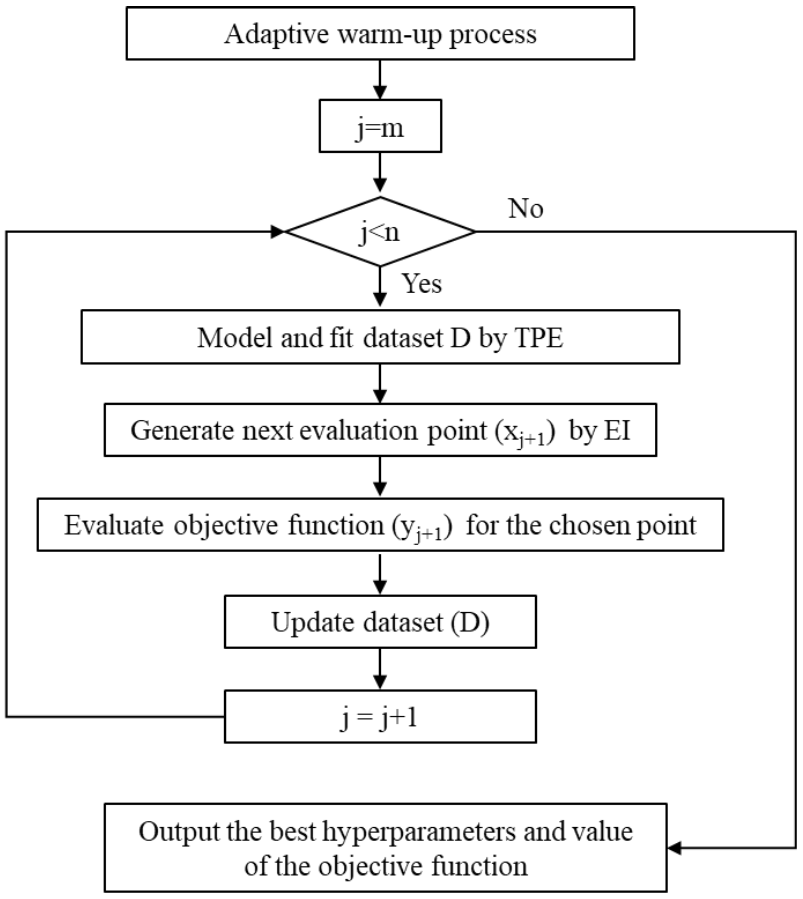

Adaptive warm-up process
j=m
j<n
No
Yes
Model and fit dataset D by TPE
Generate next evaluation point (x_{j+1}) by EI
Evaluate objective function (y_{j+1}) for the chosen point
Update dataset (D)
j = j+1
Output the best hyperparameters and value of the objective function

Figure D.2 Work for ATPE

An adaptive warm-up process is used for TPE to generate the initial evaluation set D in the case of ATPE[52]. The warm-up process adjusts the interval width (w_n) for hyperparameters according to equation D.3.

$$w_n = \begin{cases} w_n * \dfrac{1}{1+0.075n} & ,if\ y_{new} \le \dfrac{(y_{min}+y_{median})}{2} \\ w_n * \dfrac{1}{1+0.075n} * \dfrac{y_{new}}{\frac{(y_{min}+y_{median})}{2}} & ,if\ y_{new} > \dfrac{(y_{min}+y_{median})}{2} \end{cases} \qquad \text{D3}$$

where, y_{mean} and y_{median} are the mean and median of objective function values in the observation dataset, respectively. W_0 is the initial interval width, and y_{new} is the newest value of the objective function. If the updated interval exceeds the original range, the original interval width is used for the subsequent evaluation. ATPE consists of three modules: adaptive warm-up process, TPE, and EI. Figure D.3 shows the algorithm used for the adaptive warm-up process, and Figure D.2 depicts the workflow of the three modules used in ATPE.

```
Algorithm 2: Adaptive warm-up process

Input: Configuration space, Objective function, the number of points for the start-up process (m)
Output: Observation data set D; ($x_i$, $y_i$)∈ D
        1. Sample a point randomly from the configuration space
        2. Evaluate the point and add it to the data set D
for: i=1 to m
        1. Select the best point ($x_{best}$) from the dataset as the interval center
        2. Compute $y_{best}$, $y_{median}$ from D
        3. Update the interval width $w_i$. If the updated interval is beyond the original range, use the
           original width
        4. Sample one point $x_{i+1}$ from $\Theta(x_{best}, w_i)$ and evaluate it to get $y_{i+1}$
        5. Update the observation set D
end for
```

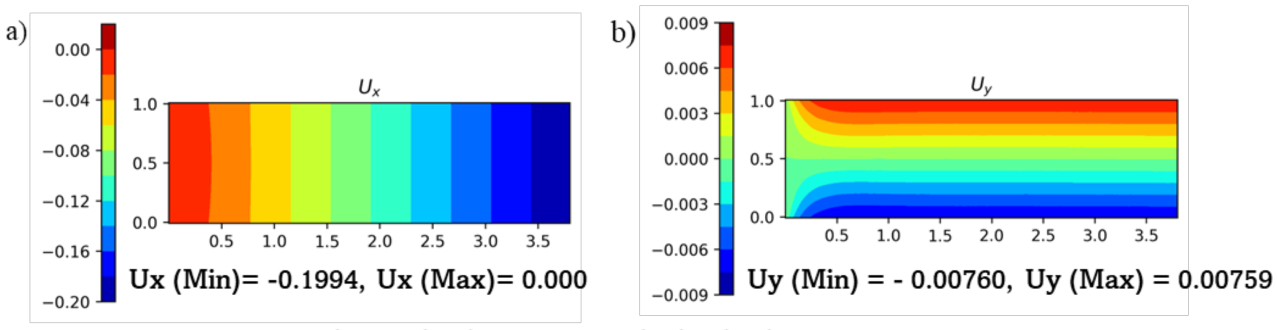
```
Return D
```

Figure D.2 Algorithm of Adaptive warm-up process for ATPE

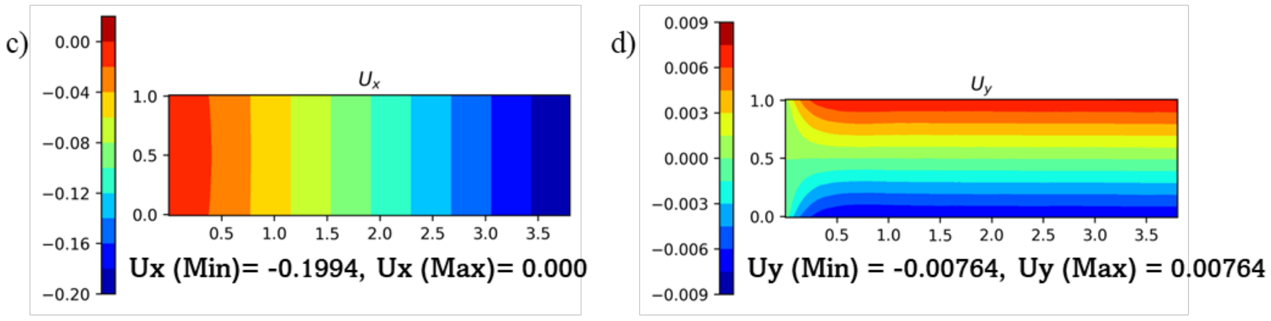
Appendix E: Displacements and errors for optimal hyperparameters

Hyperparameters optimized under compression boundary conditions.

Using optimal parameters obtained using TPE



Using optimal parameters obtained using ATPE



Using optimal parameters obtained using Random selection

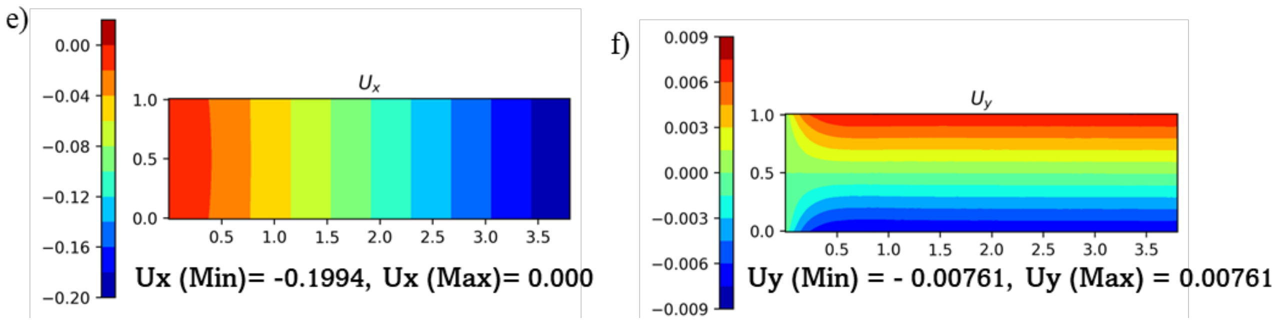


Figure E-1: Predicted displacements in x and y directions for different optimal hyperparameters obtained when uniform compression load is applied
a-b Predicted displacements under compressive loading for hyperparameters obtained using the TPE algorithm.
c-d Predicted displacements under compressive loading for hyperparameters obtained using the ATPE algorithm.
e-f Predicted displacements under compressive loading for hyperparameters obtained using Random search

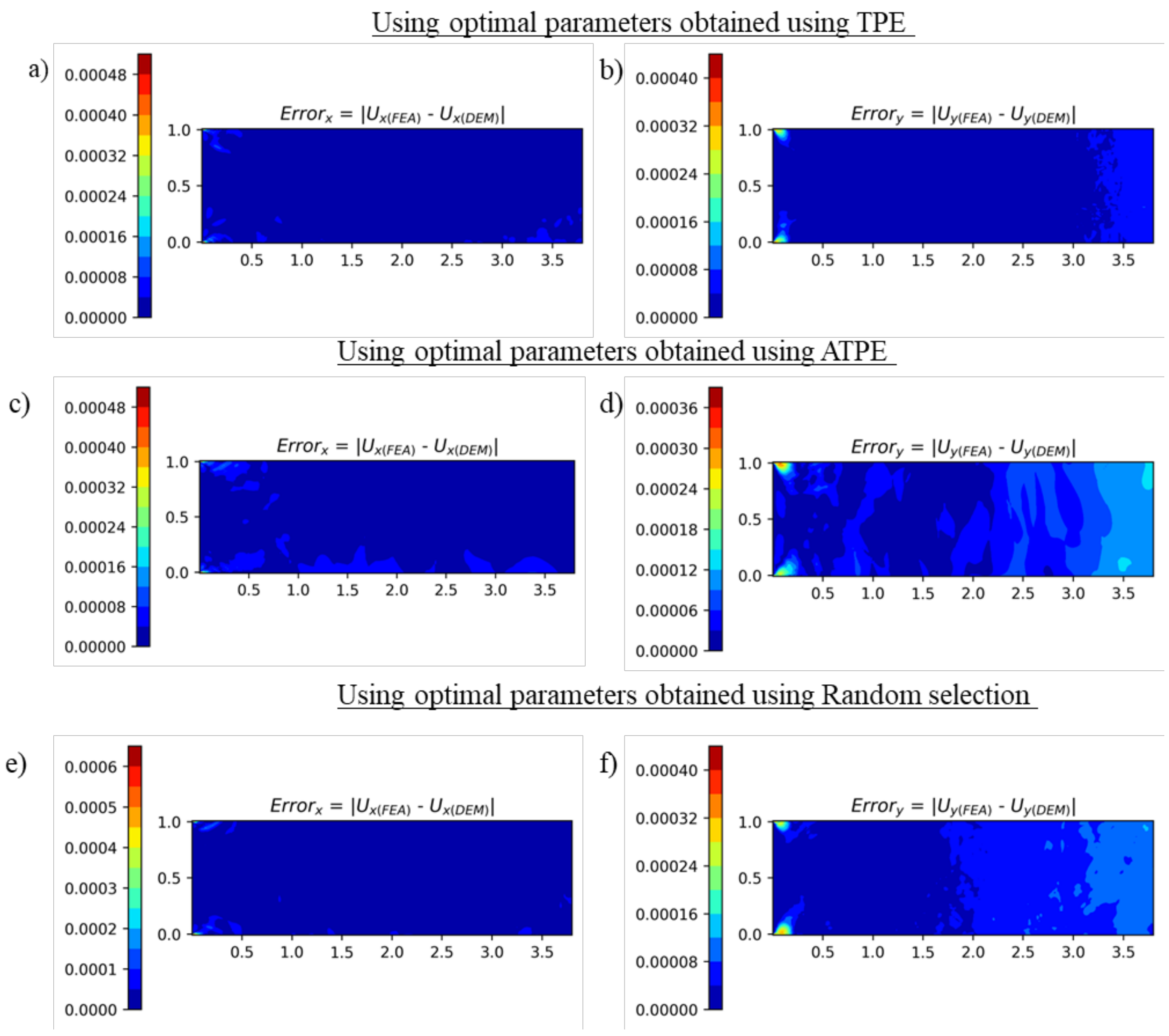


Figure E-2: Error in predicted displacements in x and y directions for different optimal hyperparameters obtained when uniform compression load is applied

a-b Error in prediction for hyperparameters obtained using the TPE algorithm.

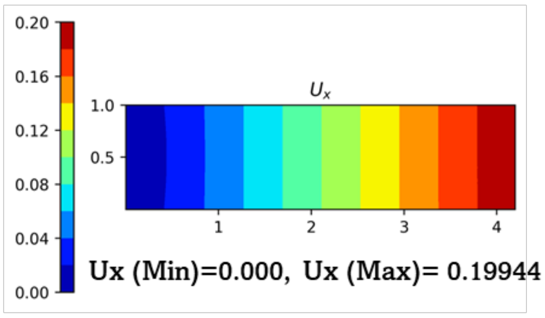
c-d error in prediction for hyperparameters obtained using the ATPE algorithm.

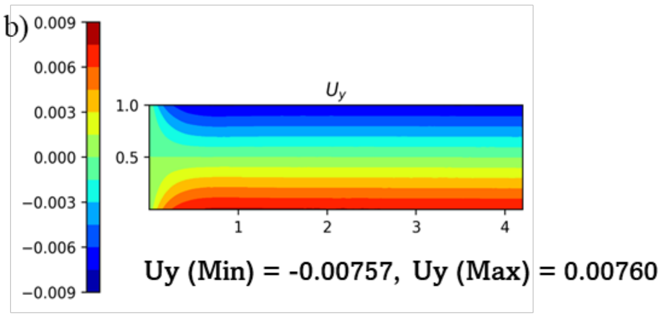
e-f Error in prediction for hyperparameters obtained using Random search

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 June 2022 doi:10.20944/preprints202206.0414.v1

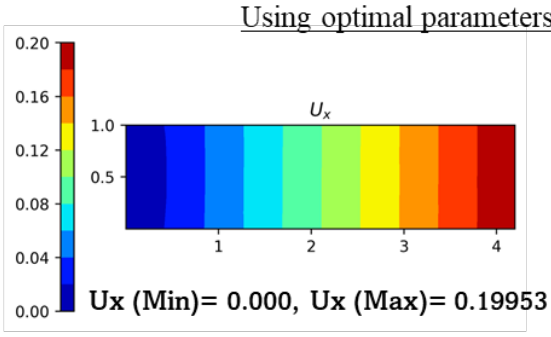
Hyperparameters optimized under tensile loading.

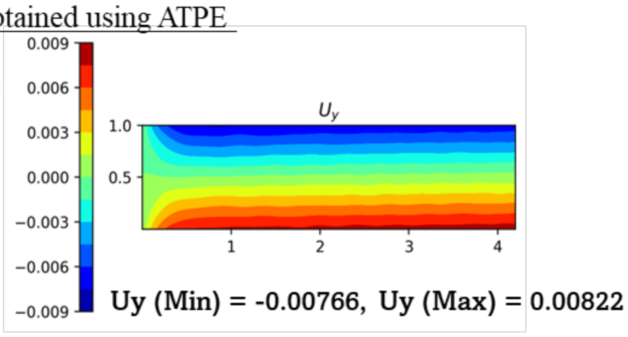
Using optimal parameters obtained using TPE

a)


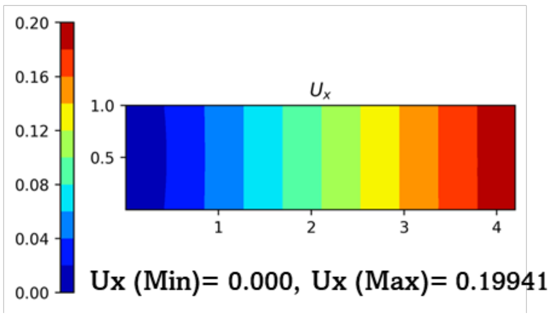
b)


Using optimal parameters obtained using ATPE

c)


d)


Using optimal parameters obtained using Random selection

e)


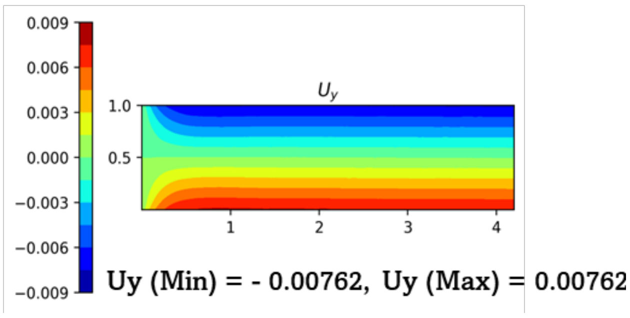
f)


Figure E3: Predicted displacements in x and y directions for different optimal hyperparameters obtained when uniform tensile load is applied.
a-b Predicted displacements under tensile loading for hyperparameters obtained using the TPE algorithm.
c-d Predicted displacements under tensile loading for hyperparameters obtained using the ATPE algorithm.
e-f Predicted displacements under tensile loading for hyperparameters obtained using Random search

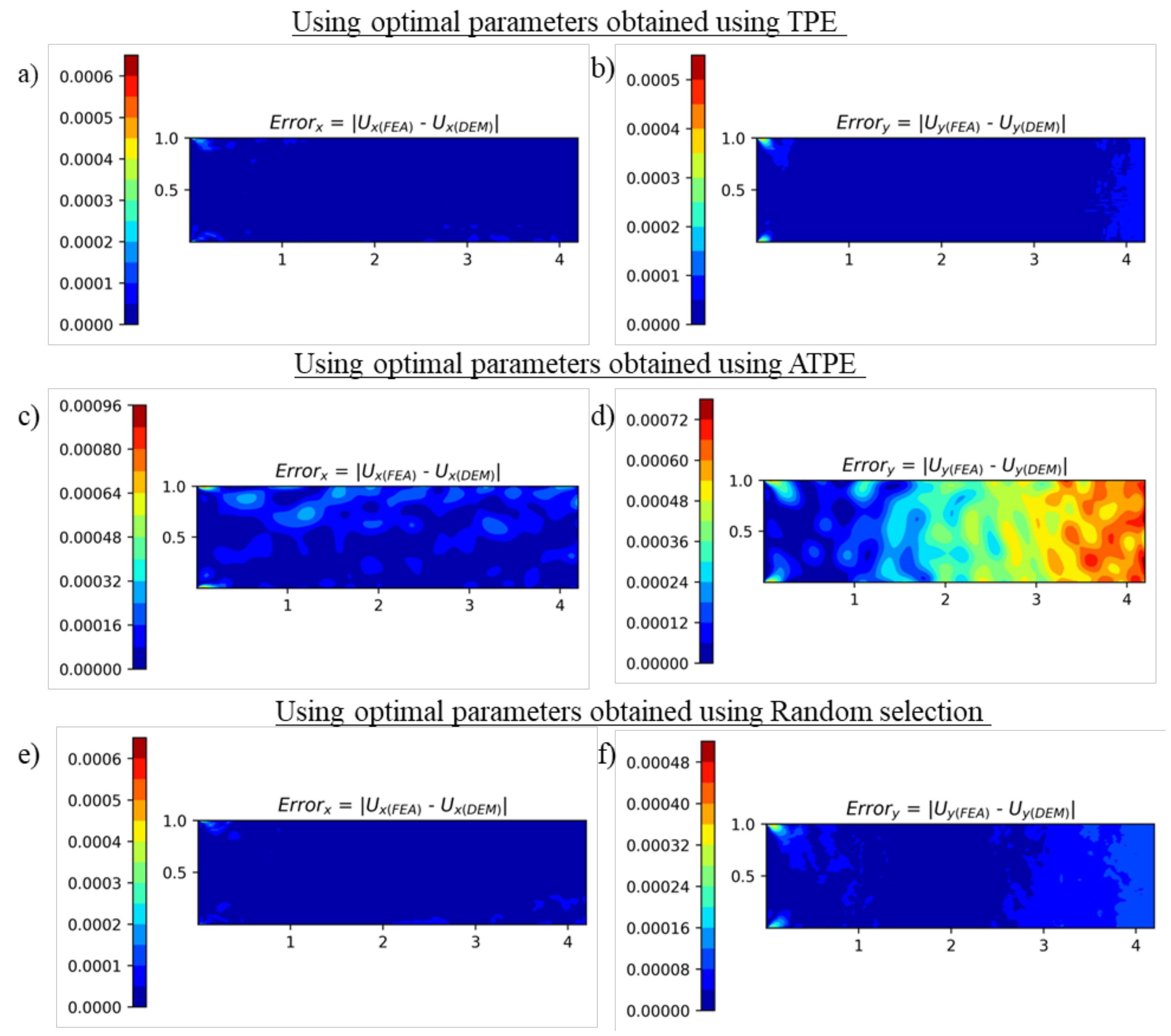


Figure E-4: Error in predicted displacements in x and y directions for different optimal hyperparameters obtained when uniform tensile load is applied

a-b Error in prediction for hyperparameters obtained using the TPE algorithm.

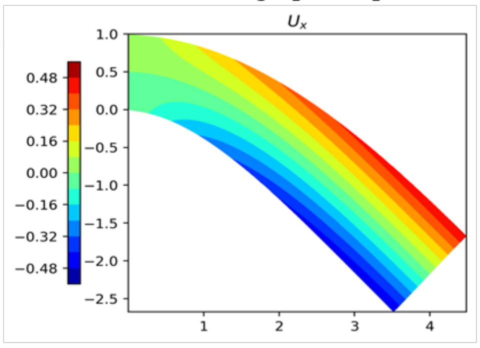
c-d error in prediction for hyperparameters obtained using the ATPE algorithm.

e-f Error in prediction for hyperparameters obtained using Random search

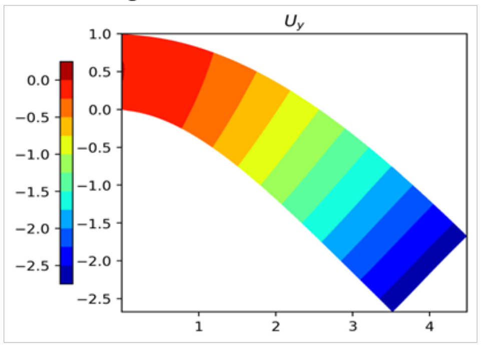
Hyperparameters optimized under bending boundary conditions.

Using optimal parameters obtained using TPE

a)



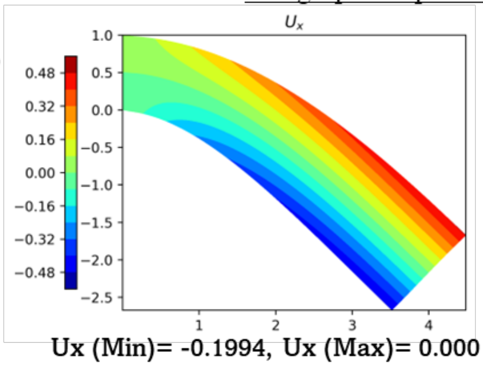
b)



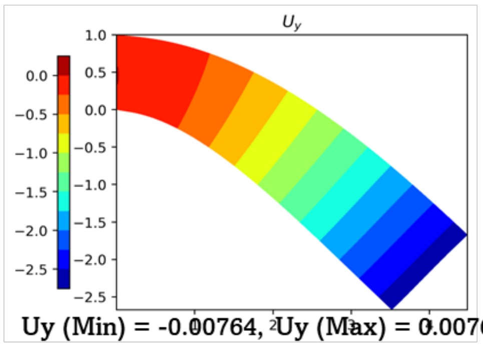
Ux(Min)= -0.48282, Ux(Max)= 0.48273 Uy (Min) = -2.67375, Uy (Max) = 0.00009

Using optimal parameters obtained using ATPE

c)



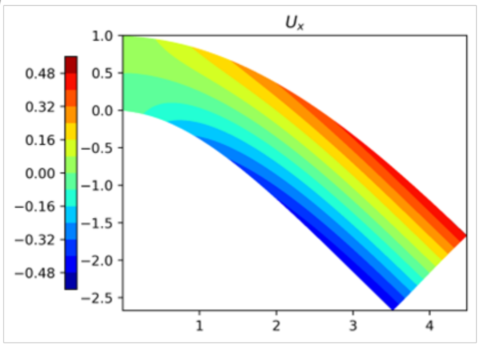
d)



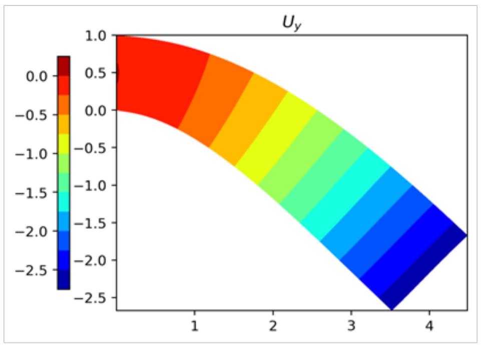
Ux (Min)= -0.1994, Ux (Max)= 0.000 Uy (Min) = -0.00764, Uy (Max) = 0.00764

Using optimal parameters obtained using Random selection

e)



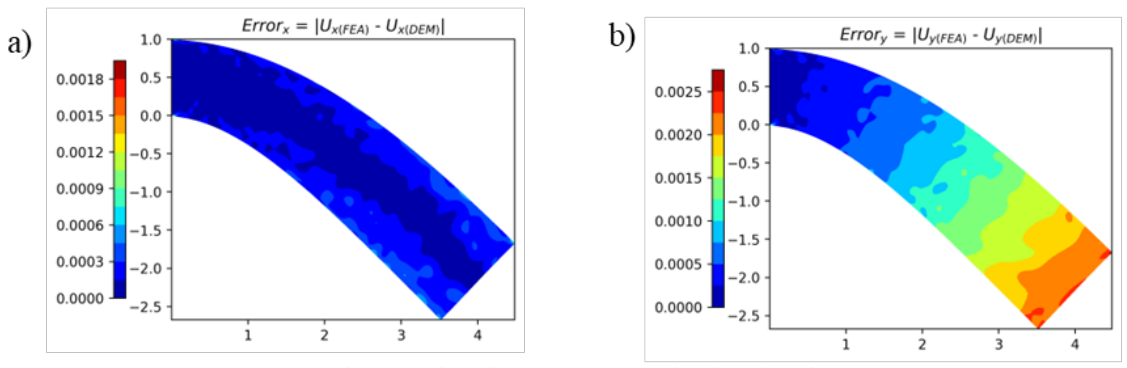
f)



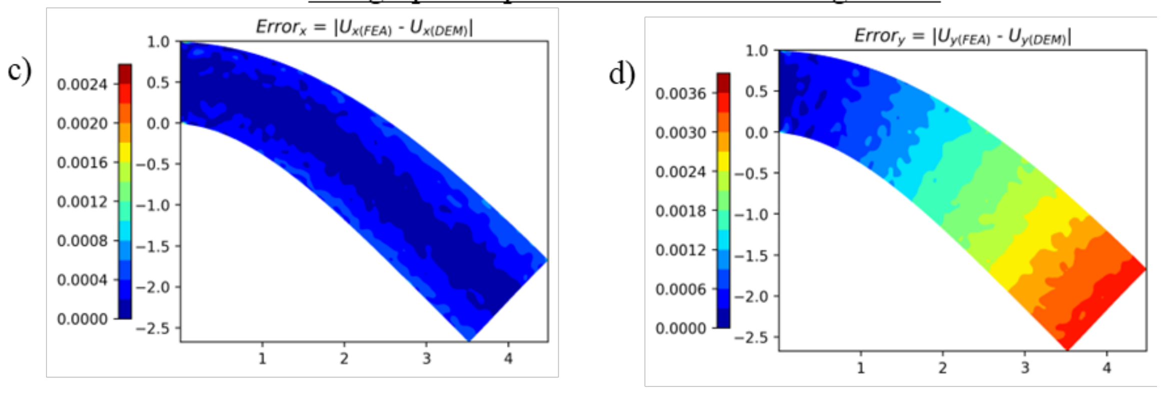
Ux (Min)= -0.19941, Ux (Max)= 0.000 Uy (Min) = - 0.00762, Uy (Max) = 0.00762

Figure E5: Predicted displacements in x and y directions for different optimal hyperparameters obtained when uniform bending load is applied.
a-b Predicted displacements under bending loading for hyperparameters obtained using the TPE algorithm.
c-d Predicted displacements under bending loading for hyperparameters obtained using the ATPE algorithm.
e-f Predicted displacements under bending loading for hyperparameters obtained using Random search

Using optimal parameters obtained using TPE



Using optimal parameters obtained using ATPE



Using optimal parameters obtained using Random selection

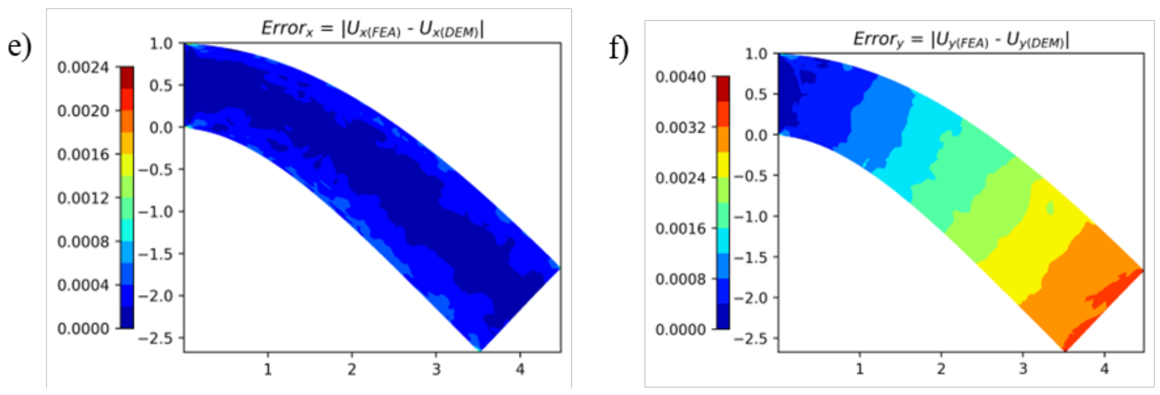


Figure E-6: Error in predicted displacements in x and y directions for different optimal hyperparameters obtained when uniform bending is applied
a-b Error in prediction for hyperparameters obtained using the TPE algorithm.
c-d error in prediction for hyperparameters obtained using the ATPE algorithm.
e-f Error in prediction for hyperparameters obtained using Random search