Abstract This paper explores the possibilities of applying physics-informed neural networks (PINNs) in topology optimization (TO) by introducing a fully self-supervised TO framework based on PINNs. This framework solves the forward elasticity problem by the deep energy method (DEM). Instead of training a separate neural network to update the density distribution, we leverage the fact that the compliance minimization problem is self-adjoint to express the element sensitivity directly in terms of the displacement field from the DEM model. Thus, no additional neural network is needed for the inverse problem. The method of moving asymptotes is used as the optimizer for updating density distribution. The implementation of Neumann, Dirichlet, and periodic boundary conditions is described in the context of the DEM model. Three numerical examples are presented to demonstrate framework capabilities: (i) compliance minimization in 2D under different geometries and loading, (ii) compliance minimization in 3D, and (iii) maximization of homogenized shear modulus to design 2D metamaterial unit cells. The results show that the optimized designs from the DEM-based framework are very comparable to those generated by the finite element method and shed light on a new way of integrating PINN-based simulation methods into classical computational mechanics problems.

1 Introduction

Neural networks (NNs) have seen wide applications in computational mechanics. They have been used to learn the complex deformation process of binary composites [1–3], thin-walled shell structures [4,5], plasticity [6–8], and constitutive laws [9–11] from a large collection of finite element simulations training sets. These models seek to approximate the underlying relations embedded in finite element solutions but do not seek to solve the underlying partial differential equations (PDEs) representing the governing physics principles. In contrast, physics-informed neural networks (PINNs) [12–15] present an alternative to the finite element method (FEM) to numerically solve the governing PDEs directly.

PINNs construct an NN approximation to the PDE solution at discrete points, and the loss function is defined based on how well the underlying physics is satisfied at these training points. One such example of PINN is the deep collocation method [13,16–18], which is based directly on the strong form of the governing PDE and seeks to minimize a loss function defined as the $L_2$ norm of the residual in the PDE at discrete points. Since it is based on the strong form, naturally, it involves the computation of second-order (e.g., in elasticity, heat transfer, and the Navier–Stokes equations) or higher-order (e.g., plate bending) spatial gradients of the...
outputs by automatic differentiation of the NN, which can be quite expensive. However, the strong-form-based frameworks are relatively straightforward to implement and are applicable to all PDEs, as they mainly involve numerically evaluating the differential operator in the PDE. Therefore, strong-form-based PINNs have been applied to solve a wide variety of problems, such as in linear elasticity, hyperelasticity, plasticity, composite mechanics [19,20], micromechanics [15], and fluid mechanics [14,21].

An alternative method to avoid calculating a high-order gradient is to write the governing equation into a variational form by finding a functional whose variation yields the governing PDE; this is termed the deep Ritz method [22,23]. For some physical systems, the governing physics dictates that the solution of the PDE yields the minimum of the energy functional, such as the principle of minimum potential energy in structural mechanics [24,25]. In these cases, it is natural to define the energy functional as the loss function, as the training of the NN is a process where the loss value is minimized. This energy-based method is termed the deep energy method (DEM) [8,26,27]. Although DEM avoids working directly with the strong form of the PDE, it is only limited to physical systems where the solution is given by a minimum potential, which is not applicable in cases like fluid mechanics. Additionally, researchers have proposed and implemented a mixed formulation, where both the strong form and the energy method are used to capture the mechanical response with high solution gradients and stress concentrations [28–30].

NNs are also widely used in topology optimization (TO) applications. NNs have been trained on optimized designs generated by finite-element TO simulations to rapidly predict optimized structures for new loading and boundary conditions [31–34] and are used as a way to parameterize the TO design space as an alternative to the element-based parameterization in FEM-based TO [35–37]. Recently, the work by Zehnder et al. [38] outlined a new way of applying PINNs to TO, where the forward elasticity problem is solved via a DEM-based technique, and another neural network is trained for the inverse design problem to generate the density-based gradient and update the design density. This is a fully self-supervised approach, where no inputs from finite-element simulations are needed. Although the training time is longer compared to FEM solution time, the NN-based TO framework generated designs that have similar, if not better, performance than FEM-based TO.

Inspired by the work of Zehnder et al. [38], we introduce a simpler, yet fully self-supervised approach for TO based on the DEM method to shed light on how PINN-based simulation techniques can be combined with traditional solution techniques in computational mechanics to provide novel solutions to classical problems. This paper is organized as follows: Sect. 2 presents an overview of deep neural networks, deep energy method, and topology optimization. Section 3 presents and discusses the results from three numerical examples. Section 4 summarizes the outcomes and limitations of this current study and highlights possible future works.

2 Methods

2.1 Deep neural networks

Deep neural networks (DNNs) consist of multiple layers of interconnected neurons, and the layers themselves are also connected. The configuration and function of DNNs bear resemblance to that of a human brain. Figure 1 shows a typical structure of a fully connected neural network model.

The architecture of a DNN can be characterized by the number of hidden layers and the number of neurons in each hidden layer. The neurons of consecutive layers are connected by a set of weights $W$ and biases $b$. The output $y^i$ of layer $i$ can be related to its corresponding weights and biases as:

$$y^i = f^i_{\text{act}}(W^i y^{i-1} + b^i),$$

where $f^i_{\text{act}}$ denote the activation function for this layer. Training of the DNN refers to the iterative process where the weights and biases of the model are updated using gradient descent [39] such that the value of a user-defined loss function is minimized.

2.2 The deep energy method

In this Section, we describe DEM in the context of the elasticity equation in structural mechanics. We consider a homogeneous, isotropic, linear elastic body undergoing small deformation. We remark that DEM is not
limited to linear elastic materials and has been applied in nonlinear hyperelastic material models [27]. In the absence of any body and inertial forces, the equilibrium equation in tensorial form reads:
\[ \nabla \cdot \sigma = 0, \quad \forall X \in \Omega. \tag{2} \]

The system is subject to Dirichlet boundary conditions:
\[ u = \bar{u}, \quad \forall X \in \partial \Omega_u, \tag{3} \]
and/or Neumann boundary conditions:
\[ \sigma \cdot n = \bar{t}, \quad \forall X \in \partial \Omega_t. \tag{4} \]

In the small deformation setting, the strain tensor is given by:
\[ \epsilon = \frac{1}{2} (\nabla u + \nabla u^T). \tag{5} \]

Stress can be computed from the constitutive law as:
\[ \sigma = \frac{E}{1 + \nu} \epsilon + \frac{E \nu}{(1 + \nu)(1 - 2\nu)} \text{tr} (\epsilon) I, \tag{6} \]
where \( E \) and \( \nu \) are Young’s modulus and Poisson’s ratio. DEM seeks the solution to the equilibrium equation via the principle of minimum potential energy. For a body in static equilibrium with no applied body forces, the potential energy of the system reads:
\[ \psi (u) = \frac{1}{2} \int_{\Omega} \sigma : \epsilon \, dV - \int_{\partial \Omega_t} \bar{t} \cdot u \, dA. \tag{7} \]

The loss function \( \mathcal{L} \) in DEM is defined identically as the potential of the system:
\[ \mathcal{L} (u) = \psi (u). \tag{8} \]

The training of the DEM model \( \mathcal{M} : \mathbb{R}^n \to \mathbb{R}^n / X \Rightarrow \tilde{u} (X) \) \(^1\) can be viewed as minimizing the loss function (system potential energy) \( \mathcal{L} \) by an optimizer. The solution to the elasticity problem, as given by the DEM model, is defined as:
\[ u^* = \arg \min_u \mathcal{L} (u). \tag{9} \]

In this work, the L-BFGS optimizer [40] was used to train the model. The learning rate is another parameter that can be changed by the user to achieve optimal framework performance. Similar to the early stopping

\(^1\) Note that the immediate output of DEM model is \( \tilde{u} \), which is different from \( u \) in Eq. (5) and Eq. (7).
schedule commonly used in model training, we monitor the relative change of the loss function and stop the training once the relative change in loss function value is less than the user-specified tolerance $\epsilon_{\text{tol}}$.

The underlying architecture of the DEM model is a fully connected feed-forward network (a multilayer perceptron model, or MLP), which is similar to the one depicted in Fig. 1 and was described in the work of Nguyen et al. [27]. Fourier transform was applied using the package Random Fourier Features Pytorch (RFF) [41] to the input features to transform them to the frequency domain, which was shown to increase the accuracy of PINNs in the work of Wang et al. [42]. The NN architecture is fully parameterizable and is characterized by the number of layers, the number of neurons (per layer), the activation function, and two standard deviation values used to initialize the weights and biases of the MLP and RFF (denoted by $\sigma_{\text{MLP}}$ and $\sigma_{\text{RFF}}$, respectively). Hyperparameter optimization can be used to fine-tune and improve the prediction accuracy of the model; see the work of Chadha et al. [43] for a detailed description of the optimization framework architecture and examples.

2.3 Enforcement of boundary conditions

In the context of elasticity problems, three different types of boundary conditions (BCs) are common, namely the Neumann, Dirichlet, and periodic BCs. Enforcement of Neumann BCs is straightforward. The value of the prescribed traction $\mathbf{t}$ enters directly in the boundary integral part of Eq. (7), while the boundary displacements are provided by the DEM model. Traction-free surfaces are satisfied trivially as the boundary integral vanishes on these surfaces. A surface (in 3D) or edge (in 2D) integration rule needs to be defined for numerically evaluating the surface integral in regions where traction is prescribed.

Dirichlet BCs are typically enforced via the penalty method by adding a penalty term to the loss function [17,27,44]:

$$\text{MSE}_{\partial \Omega_u} = \frac{1}{N_u} \sum_{j=1}^{N_u} ||\mathbf{u} - \bar{\mathbf{u}}||^2,$$

leading to a modified loss function definition:

$$\mathcal{L}(\mathbf{u}) = \psi(\mathbf{u}) + w \text{MSE}_{\partial \Omega_u},$$

where $w$ is a user-defined weight parameter. To avoid defining the additional parameter $w$, we adopted a direct approach to enforce Dirichlet BCs via an additive decomposition [45]:

$$\mathbf{u}(X) = \mathbf{\tilde{u}}(X) \ast \mathbf{m}(X) + \mathbf{u}_0(X),$$

where $\ast$ denotes element-wise multiplication between two vectors. The continuous functions $m$ and $u_0$ are chosen such that:

$$\mathbf{m}(X) = \mathbf{0} \text{ and } \mathbf{u}_0(X) = \mathbf{\tilde{u}}, \forall X \in \partial \Omega_u$$

where $\mathbf{m}(X)$ denotes the value of the function $m$ evaluated at discrete points $X$ (similarly for $u_0$). By design, $\mathbf{u}(X)$ satisfies all Dirichlet boundary conditions for all arbitrary $\mathbf{\tilde{u}}$ produced by the DEM model. Since the DEM generates a continuous and differentiable raw displacement field, and the functions $m$ and $u_0$ are continuous, the final displacement field as computed by Eq. (12) is also continuous and differentiable.

Periodic BCs are commonly used when simulating a representative volume element that repeats infinitely in space. It is also used when simulating unit cells of periodic metamaterials. In the context of the FEM, periodic BCs are typically stated in the form of constraint equations [46]:

$$u^k_+ - u^k_- = \epsilon^0_{ij} \left(x^k_j - x^k_{-j}\right),$$

where $k-$ and $k+$ denote the left and right instances of the $k^{th}$ periodic boundary pair, and $\epsilon^0_{ij}$ is the (known) applied strain tensor. For easier convergence of the DEM training process, we adopted a mixed implementation for periodic BCs, where we treat zero and nonzero applied strain components differently. The boundary displacement induced by a nonzero applied strain component is enforced as a pair of Dirichlet BCs, and zero
applied strain components are enforced by penalty. Without loss of generality, let the nonzero component of $\epsilon_{ij}^0$ be $\epsilon_a$, which is applied on the $k^{th}$ periodic boundary pair. This leads to two Dirichlet boundary conditions:

$$u(X) = 0, \ \forall X \in \partial \Omega_{k-},$$
$$u(X)_a = \Delta L \epsilon_a, \ \forall X \in \partial \Omega_{k+},$$

(15)

where $\Delta L$ is the distance between the $k^+$ and $k^-$ boundary pair. Note that Eq. (15) is a simplification of Eq. (14) since, in general, the displacement on $\partial \Omega_{k-}$ does not need to be 0, provided that it satisfies Eq. (14). This simplification was made to ensure that the applied strain is enforced exactly by Dirichlet BCs. For all other boundary pairs that correspond to zero applied strain components, the constraint Eq. (14) reduces to:

$$u_i^{k+} = u_i^{k-}.$$  

(16)

Therefore, the following penalty term is added to the loss function:

$$\text{MSE}_{\text{PBC}} = \frac{E}{N_p} \sum_{j=1}^{N_p} ||u_i^{k+} - u_i^{k-}||^2,$$

(17)

where $E$ is Young’s modulus of the material (a physics-informed penalty weight) and $N_p$ is the number of points of the $k$th periodic boundary. Note that due to the minimization nature of the training process there is no guarantee that the added boundary loss term will be exactly 0 at the end of the DEM training process. That is, the constraints enforced by a penalty loss are not exact. Therefore, the value of the penalty weight can be optimized as an additional hyperparameter to ensure that the constraints are better enforced. However, this is outside of the scope of this work, and we used instead a simple estimate ($E$), which comes from the stiffness of the parent material.

2.4 Topology optimization

In this Section, we describe topology optimization (TO) in the context of DEM. The mathematical formulation of the TO problem using DEM can be stated as:

$$\min_{\rho} f(u, \rho)$$
$$\text{s.t. } u = \arg \min_u L(u, \rho),$$
$$V_e \sum \rho = \bar{V},$$
$$0 \leq \rho_e \leq 1, \ \forall e = 1 \cdots N,$$

(18)

where $f(u, \rho), V_e$ and $\bar{V}$ denote the objective function to minimize, element density vector, element volume, and target volume, respectively. A typical objective of TO is to minimize the system compliance subject to a volume/mass constraint. For compliance minimization, the objective function is:

$$f_{\text{comp}}(u, \rho) = \frac{1}{2} \int_{\Omega} \sigma^*(\rho) : \epsilon \ \text{d}V$$

(19)

where $\sigma^*(\rho)$ is the stress computed from the classical solid isotropic material penalization (SIMP) scheme [47]:

$$\sigma^*(\rho) = \rho^p \star \sigma.$$  

(20)

The penalty exponent $p$ is taken to be 3 in this work.

For designing periodic 2D meta structures, a commonly employed objective is to maximize the homogenized shear modulus of the unit cell [31,46,48]:

$$G_{\text{homo}} = \frac{1}{A} \int_{\Omega} \sigma^p(\rho) : \epsilon^p \ \text{d}A,$$

(21)
where $A$ is the area of the 2D element, $\sigma^p$ and $\epsilon^p$ denote the stress and strain fields computed from the periodic displacement field subjected to the following applied simple shear loading ($\epsilon^0$ in Eq. (14)):

$$\epsilon^0 = \begin{bmatrix} 0 & 0.01 \\ 0.01 & 0 \end{bmatrix}. \quad (22)$$

To fit into a minimization framework, the objective function is defined as:

$$f_{\text{shear}}(u, \rho) = -G_{\text{homo}}. \quad (23)$$

Due to the introduction of the element density vector, we correspondingly modify the definition of the potential energy and loss function as:

$$\psi(u, \rho) = \mathcal{L}(u, \rho) = \frac{1}{2} \int_{\Omega} \sigma^*(\rho) : \epsilon \, dV - \int_{\partial\Omega} \bar{t} \cdot u \, dA. \quad (24)$$

In this work, we chose to evaluate the volume integral in Eq. (24) by Gauss quadrature integration. This necessitates the formation of quadrilateral (in 2D) or hexahedral (in 3D) isoparametric finite elements from the structured nodes $X$ in the DEM domain, which is very similar to a FEM-type treatment. The need to form elements is a major disadvantage of the framework compared to other meshless DEM methods like those developed in [17,27], which only require nodes but not elements to exist in the simulation domain. The gradient operator in Eq. (5) is evaluated by the gradients of the finite element shape functions and not from the automatic differentiation of the DEM model. The density vector $\rho$ is defined at the cell center, identical in classical FEM-based TO. The method of moving asymptotes (MMA) [49] is used to iteratively update the density based on the sensitivity.

2.5 Sensitivity analysis

In this section, we describe the sensitivity analysis in the context of DEM. Comparing Eq. (19) and Eq. (23), we see that both are similar in form to a classical compliance minimization problem in FEM-based TO. Since the compliance minimization problem is self-adjoint, the adjoint vector is simply the negative of the displacement, and therefore, no additional adjoint analysis is needed to evaluate the sensitivity. In this case, one simply needs to express the sensitivity in terms of the displacement field and its gradients produced by DEM. For the compliance minimization problem Eq. (19), the sensitivity of element $e$ is given by:

$$\frac{\partial f_{\text{comp}}}{\partial \rho_e} = -\frac{1}{2} p e^p - 1 \int_{\Omega_e} \sigma : \epsilon \, dV. \quad (25)$$

Similarly, for maximization of the shear modulus in 2D meta material unit cells (i.e., Eq. (23)), the sensitivity of element $e$ is given by:

$$\frac{\partial f_{\text{shear}}}{\partial \rho_e} = \frac{1}{A_e} p e^p - 1 \int_{\Omega_e} \sigma : \epsilon \, dA. \quad (26)$$

To avoid checkerboarding and mesh dependence, the density filtering technique introduced by Bruns et al. [50] was employed. In this approach, the MMA solver does not directly manipulate $\rho$, but instead manipulates on a pseudo-density (the design variables) $\xi$ that is related to $\rho$ as:

$$\rho = W \xi. \quad (27)$$

where $W$ is the normalized filter matrix whose rows are given by the normalized weights $\tilde{q}_{ij}$:

$$q_{ij} = \max(r_{\text{min}} - ||X_i - X_j||),$$

$$\tilde{q}_{ij} = \frac{1}{\sum_{k=1}^{N} q_{ik}} q_{ij}. \quad (28)$$

Algorithm 1 summarizes the process of TO using the DEM method.
Algorithm 1: Topology optimization using the deep energy method

| Input: | Network architecture, domain size, grid size, material properties, \( C_{tol}, \gamma_{min}, \tilde{V} \), initial density distribution \( \xi^0 \), maximum TO iterations |
|---|---|
| Output: | Optimized design defined by density array \( \rho \) |

/* Initialization */
1. Build filter matrix \( W \)
2. Initialize weights and biases of the DEM model \( M \)
3. \( i \leftarrow 0 \)

/* Begin topology optimization */
4. while \( i < \text{max TO iteration} \) do
5. Apply filter: \( \rho^i = W \xi^i \)

/* Begin DEM training */
6. while not converged do
7. Obtain \( \tilde{u} \) from \( M \), apply Dirichlet BCs to get \( u \)
8. Use shape function gradient operator to compute \( \epsilon \)
9. Use \( \rho \) to compute \( \sigma^* \)
10. Compute loss
11. Update the weights and biases of \( M \) by back-propagation

/* Begin sensitivity calculation */
12. Obtain converged \( \tilde{u} \) from \( M \), apply Dirichlet BCs
13. Compute \( \epsilon \) and \( \sigma^* \)
14. Compute the element-wise filtered sensitivity
15. Invert filtering for sensitivity: \( \frac{\partial f}{\partial \xi} = W^{-1} \frac{\partial f}{\partial \rho} \)
16. Pass sensitivity to MMA solver to obtain \( \xi^{i+1} \)
17. \( i \leftarrow i + 1 \)

Table 1 Hyperparameters of the DEM model

| Layers | Neurons | Activation function | Max iteration | Learning rate \( \sigma_{\text{MLP}} \) | Learning rate \( \sigma_{\text{RFF}} \) |
|---|---|---|---|---|---|
| Value | 5 | 68 | RReLU | 100 | 1.735 | 0.0622 | 0.1192 |

3 Results and discussion

In this Section, we present three numerical examples that showcase the capabilities of our DEM-based TO framework. In the first example, we performed compliance minimization in 2D for two different geometries: 1. a long slender bridge fixed on both ends subject to center downward load, and 2. a short cantilever beam subject to downward load on its right free end. In the second example, we extend the bridge example to 3D. In the last example, we maximize the shear modulus of 2D periodic unit cells to design metamaterials and compare the results at different initial conditions. In all cases, we compared our DEM results with those obtained from FEM. All training of the DEM model was done on Delta, a high-performance computing cluster hosted at the National Center for Supercomputing Applications (NCSA). A representative GPU compute node on Delta has the following hardware specifications: one AMD Milan CPU, four NVIDIA A40 GPUs, and 256 GB RAM. The hyperparameters of the DEM model are provided in Table 1 and were obtained following the procedure described in the work by Chadha et al. [43].

The DEM model was partly adapted from the works of Nguyen et al. [27] and Chadha et al. [43], and was implemented in PyTorch (version 1.11.0) [51]. Material properties used in all three examples are: \( E = 200 \) MPa and \( \nu = 0.3 \). Plane stress condition and unit thickness were assumed for all 2D simulations. A target volume fraction of 40% was used. A Python implementation of the MMA solver by Chandrasekhar et al. [52] was used. For all DEM examples presented, 80 TO iterations were performed using a filter radius of 0.25 m.

3.1 Compliance minimization in 2D

Two different cases are presented in this example. In the first case, a 2D rectangular bridge of size 12-by-2 m\(^2\) was subjected to a downward surface traction at the center of the top edge over a length of 0.5 m. The bridge was fixed at its left and right edges, and 3751 nodes were placed into the domain forming a 121-by-31 grid. In the second case, a 2D rectangular beam of size 10-by-5 m\(^2\) was subjected to a downward concentrated...
load at the center of the right edge, while the left edge was fixed. 4186 nodes were placed into the domain, forming a 91-by-46 grid. The relative convergence tolerance $\epsilon_{tol}$ was set to $5 \times 10^{-6}$ for all DEM training. As a comparison, FEM-based TO was applied to solve the same problem using an identical node layout.

The designs from DEM- and FEM-based TO at different design iterations are compared in Figs. 2 and 3. The relative compliance reduction for both cases as the TO optimizer progressed is presented in Fig. 4a. To give a quantitative comparison of the similarity between DEM- and FEM-based designs, the dice similarity coefficient (DSC) was computed for both cases and is presented in Fig. 4b. The DSC is defined as:

$$DSC = \frac{2|I_{\text{FEM}} \cap I_{\text{DEM}}|}{|I_{\text{FEM}}| + |I_{\text{DEM}}|},$$

where $I$ represents the binarized density using a (dynamic) threshold of 0.5 times the current maximum density. Finally, the computational cost for running the two cases using DEM and FEM\(^2\) is summarized in Fig. 4c and is compared in Table 2.

From Figs. 2 and 3, we see that the DEM- and FEM-based TO frameworks produced almost identical simulation design evolution history and final designs. The similarity in designs is further demonstrated in Fig. 4b, where the DSC is larger than 90% for the final designs in both cases. For the bridge case, we noticed that the DSC remains larger than 98% for all design iterations, showing a close resemblance between the DEM- and FEM-based approaches. The two frameworks also produce designs of similar performance, as is evident in Fig. 4a, where the compliance reduction history is almost identical between the two methods due to the use of an identical MMA optimizer. Nonetheless, Table 2 shows that FEM still holds a massive advantage over DEM in terms of computational time, which agrees with the trends observed in similar DEM-based TO work [38]. However, a very tight relative tolerance of $5 \times 10^{-6}$ was used in training, which could be loosened to gain computational efficiency. From Fig. 4c, we highlight two important features of DEM-based TO. For both cases, the training time shows a decreasing trend as the design progressed. The reducing trend is due to the fact that the weights and biases from the previous TO iteration are used as the initial condition for the next iteration, a form of transfer learning, which leads to gradual training time reduction. The gradual time reduction is in sharp contrast to FEM-based TO, where the computational time for each iteration is constant and can be very

\(^2\) The highly optimized Python TO code found in [53] was used for time comparison, which was based on the MATLAB code by Andreassen et al. [54].
3.2 Compliance minimization in 3D

The second example concerns a 3D rectangular bridge of size 12-by-2-by-2 m$^3$. The structure was fixed at both end surfaces and was subjected to a downward load at the center of the top surface in a square region of size 0.5-by-0.5 m$^2$. 75625 nodes were placed into the domain, forming a 121-by-25-by-25 grid. $\epsilon_{\text{tol}}$ was set to $5 \times 10^{-5}$. As a comparison, Abaqus/Standard [55] was used to perform TO using the built-in TO functionalities and optimizer.

The final designs from DEM and FEM are presented in Fig. 5a, b, respectively. The STL file for the final FEM design was extracted using a density threshold of 0.8, and the same threshold was used to obtain the voxelated DEM final design. The two designs are overlaid and presented in Fig.5c, d. The relative compliance reduction and training time are shown in Fig. 6. The total TO time for DEM was 1050.4s.

From the overlaid plot and the section view in Fig. 5 we see that the DEM-based design largely resembles the FEM-based design, especially near both ends of the bridge. Some design differences can be seen in the center, especially from the section view. From Fig. 6a, we see that DEM was able to quickly reduce the compliance in less than 10 iterations, but has a final compliance that is larger than that of the FEM design. The evolution histories are different since we used the MMA solver to update the design density, while the reference FEM solution in this example was generated with Abaqus’s built-in optimizer. Due to the nonconvex nature of TO problems, the evolution histories depend on the optimizer used to update the design density. This process is not to be confused with the optimizers used for hyperparameter optimization and training of the DEM model, which have little effect on the different evolution histories. From Fig. 6b, we again see that training time decreased as the design progressed, a trend that is consistent with Sect. 3.1. Finally, we highlight that the same hyperparameters in Sect. 3.1 were used directly in this 3D example without any additional hyperparameter optimization. This is possible since a neural network with sufficient depth (number of hidden neurons) has the potential to approximate different complex displacement fields, irrespective of the spatial dimensions (2D in Sect. 3.1 and 3D in Sect. 3.2). The difference is that the weights and biases of the trained DEM models are different in those cases, based on the specific spatial dimensionality, loading, and geometry used to define the DEM loss function. The satisfactory results observed in this example also underscore the robustness of the DEM model with respect to hyperparameter selection.
Fig. 5 Comparison of final designs after 80 design iterations. The FEM-based design is shown in the form of an stl surface, while the DEM design is shown as voxels.

Fig. 6 DEM-based TO in 3D: a Relative compliance evolution for the 3D bridge design. b Training time for each iteration.

3.3 Shear modulus maximization in 2D

In this example, a 2D unit cell of size 10-by-10 m$^2$ was subjected to a shear load at the top edge. The unit cell was subjected to periodic BCs in the X- and Y-directions. Further, 6561 nodes were placed into the domain forming an 81-by-81 grid. The objective is to maximize the homogenized shear modulus while maintaining the volume fraction at 40%. It is well known that the final optimized design depends significantly on the initial material distribution [31,46,48]. Therefore, three different initial designs were considered with different initial hole diameters, as shown in the first row of Fig. 7. $\epsilon_{tol}$ was set to $5 \times 10^{-6}$. For comparison, the DEM-based TO results are compared to their FEM-based counterparts produced by the MATLAB [56] code developed by Xia et al. [46].

The initial and final designs from DEM- and FEM-based TO are compared in Fig. 7. The relative change in shear modulus for both cases over all the iterations is presented in Fig. 8a. Further, DSC coefficient is calculated for the binarized final designs obtained from both cases using Eq. (29) and is presented in Fig. 8b. The DEM training time for the three cases was 579.7, 934.4, and 1125.7s, respectively. Training time is summarized in Fig. 8c.
Fig. 7 Density plot of designs generated by DEM and FEM for 2D shear modulus maximization. The first row shows three different initial configurations, and the second and third rows show the corresponding final designs.

From Fig. 7, we see that the DEM- and FEM-based TO frameworks produced very similar final designs for all three initial designs. However, comparing Fig. 7f, i, we noticed an important deficiency of the DEM-based TO method. The final FEM-based TO design (Fig. 7i) has fourfold symmetry, while the DEM-based design (Fig. 7f) does not. In fact, the DEM-based final design is not periodic at the lower left corner. This difference is due to the fact that periodic BCs are enforced in FEM-based simulations by constraining the corresponding degrees of freedom of the global stiffness matrix. Hence, periodicity of the displacement field (hence the material distribution) is guaranteed. Also, if the initial design is symmetric, symmetry persists in subsequent designs. This is not the case for DEM-based TO, as the periodic BCs were enforced by penalty. The periodic penalty loss gradually crept up as the design progressed, and the designs gradually lost periodicity due to minor violations of periodicity in the displacement field and material distribution. Nonetheless, it is observed that the DSC is larger than 90% for the final designs in all three cases. The low DSC during the initial stage of
Fig. 8 Comparing DEM-based TO with FEM-based TO in 2D shear modulus maximization: a relative shear modulus evolution for the two cases. b Dice similarity coefficient computed on the binarized designs. c Training time for each iteration

evolution results from the different optimizers used in the DEM- and FEM-based TO code. The DEM-based TO used the MMA optimizer for design density update, while the reference FEM code in this example used the optimality criteria method, which is another commonly utilized optimizer in TO problems for density update. Similar to the two examples presented above, we again see a gradual reduction of training time as the optimizer progressed.

4 Conclusions and future work

This work uses the DEM-based simulation technique in TO. The proposed framework combines solutions to PDEs using PINNs and classical TO, where a deep neural network using DEM solves the momentum balance equation by minimizing the system’s potential energy. Instead of training a separate neural network to predict the updated density array as in the work of Zehnder et al. [38], we leverage the fact that the compliance minimization problem is self-adjoint and express element sensitivity directly using the displacement field produced by the DEM model. Therefore, only one neural network needs to be trained in each optimization iteration, making the framework much simpler. The training of the DEM model also leverages transfer learning to reduce training time, where the weights and biases from the previous iteration are used as the initial condition for the next iteration, which leads to a significant reduction of training time in later iterations. The proposed framework combines two different types of optimizers: an L-BFGS optimizer for the DEM model training and a classical MMA optimizer for updating the density array.

Three numerical examples are presented to demonstrate the capabilities of the proposed framework. In the first example, we performed compliance minimization in 2D on two different loading and boundary conditions. We saw that the DEM-based topology optimization framework generated almost identical designs to its FEM counterpart. In the second example, we extended compliance minimization to 3D, where we optimized a 3D bridge under transverse loading. We found that the hyperparameters used in 2D simulations were effective in 3D as well, generating a design quite similar to the one generated by Abaqus. In the last example, we attempted the design of 2D metamaterials by maximizing the homogenized shear modulus, which necessitated the implementation of periodic boundary conditions in DEM. For three different initial designs, our DEM framework was able to generate designs very similar to FEM, albeit suffered from a minor loss of symmetry due to the inaccurate enforcement of periodic boundary conditions. In all examples presented, the training time for the DEM model was longer than the solution time for FEM. Still, training time shows a gradual decrease with design iterations and is relatively insensitive to the number of nodes in the simulation domain. Thus, the DEM-based TO method is potentially beneficial when a fine mesh and many TO iterations are required.

We conclude that the proposed DEM-based topology optimization framework can produce optimized designs very similar to FEM. Although having a high computational cost, we emphasize that the goal of this work is not to outperform FEM in particular applications but rather to highlight the diverse application scenarios for the DEM method and shed light on how neural-network-based solutions for PDEs can be employed in engineering applications.

A primary limitation of this work is the need to form element discretization of the domain and use shape function gradients to compute gradients of field variables produced by DEM, which is in contrast to many previous studies [17,38,44] that used a meshless method to solve the underlying governing equation. Extension
to a meshless formulation using automatic differentiation of the DEM model will be our future work. We will also explore topology optimization for hyperelastic materials undergoing finite deformation, where Newton–Raphson iterations need to be employed in FEM, rendering it rather expensive. In all examples presented in this work, we only considered the volume constraint. In practice, this is often supplemented with stress constraints to avoid failure of the designed structure. A stress constraint can also be incorporated into the DEM-TO framework (one needs to express the sensitivity of the aggregated stress constraint in terms of the stress and strain at each finite element as computed by the trained DEM model) and will be our future work. In addition, the effects of enhancing the loss function with its gradient [12, 57] on the DEM model accuracy will be studied.

Acknowledgements The authors would like to thank the National Center for Supercomputing Applications (NCSA) for providing the computational resources on the Delta cluster.

Author contributions JH did conceptualization, methodology, software, formal analysis, investigation, data curation, writing—original draft. CC performed software, formal analysis, investigation, writing—original draft. SK was involved in software, formal analysis, investigation, writing—original draft. SK was involved in supervision, resources, writing—review & editing. DA done supervision, resources, writing—review & editing. IJ contributed to supervision, resources, writing—review & editing, funding acquisition.

Data availability The data and source code that support the findings of this study can be found at: https://github.com/Jasiuk-Research-Group/DeepEnergy-TopOpt.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Chen, C.-T., Gu, G.X.: Machine learning for composite materials. MRS Commun. 9(2), 556–566 (2019)
2. Yang, C., Kim, Y., Ryu, S., Gu, G.X.: Prediction of composite microstructure stress–strain curves using convolutional neural networks. Mater. Des. 189:108509 (2020)
3. Luo, L., Zhang, B., Zhang, G., Li, X., Fang, X., Li, W., Zhang, Z.: Rapid prediction and inverse design of distortion behaviors of composite materials using artificial neural networks. Polym. Adv. Technol. 32(3), 1049–1060 (2021)
4. He, J., Kushwaha, S., Abueidda, D., Jasiuk, I.: Exploring the structure-property relations of thin-walled, 2d extruded lattices using neural networks. Comput Struct. (2022), in press. https://doi.org/10.1016/j.compstruct.2022.106940
5. Chen, G., Li, T., Chen, Q., Ren, S., Wang, C., Li, S.: Application of deep learning neural network to identify collision load conditions based on permanent plastic deformation of shell structures. Comput. Mech. 64(2), 435–449 (2019)
6. Stoffel, M., Bamer, F., Markert, B.: Neural network based constitutive modeling of nonlinear viscoplastic structural response. Mech. Res. Commun. 95, 85–88 (2019)
7. Maysam B, Gorji, Mojtaba, Mozaffar, Julian N, Heidenreich, Jian, Cao, and Dirk, Mohr: On the potential of recurrent neural networks for modeling path dependent plasticity. J. Mech. Physics Solids, 143: 103972 (2020)
8. Diab, W., Abueidda, Seid, Koric, Nahl A, Sobh, and Huseyin, Sehitoglu. Deep learning for plasticity and thermo-viscoplasticity. Int. J. Plasticity, 136:102852 (2021)
9. Chen, G.: Recurrent neural networks (rnn) learn the constitutive law of viscoelasticity. Comput. Mech. 67(3), 1009–1019 (2021)
10. Yang, H., Xiang, Q., Tang, S., Guo, X.: Learning material law from displacement fields by artificial neural network. Theor. Appl. Mech. Lett. 10(3), 202–206 (2020)
11. Flaschel, M., Kumar, S., De Lorenzis, L.: Unsupervised discovery of interpretable hyperelastic constitutive laws. Comput. Methods Appl. Mech. Eng. 381, 113852 (2021)
12. Khemraj, Shukla, Mengjia, Xu, Nathaniel, Trask, and George E, Karniadakis: Scalable algorithms for physics-informed neural and graph networks. Data-Centric Engineering, 3 (2022)
13. 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. 379, 113741 (2021)
14. Shengze, Cai, Zhiping, Mao, Zhicheng, Wang, Minglang, Yin, and George Em, Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. Acta Mech. Sinica, 1–12 (2022)
15. Henkes, A., Wessels, H., Mahnken, R.: Physics informed neural networks for continuum micromechanics. Comput. Methods Appl. Mech. Eng. 393, 114790 (2022)
16. Raissi, M.: Deep hidden physics models: Deep learning of nonlinear partial differential equations. J. Machine Learning Res 19(1), 932–955 (2018)
17. Diab, W., Abueidda, Qiyue, Lu, and Seid, Koric: Meshless physics-informed deep learning method for three-dimensional solid mechanics. Int. J. Numerical Methods Eng. 122(23):7182–7201 (2021)
18. Hongwei, Guo, Xiaoying, Zhuang, and Timon, Rabczuk: A deep collocation method for the bending analysis of Kirchhoff plate. arXiv preprint arXiv:2102.02617 (2021)
19. Yan, C.A., Vescovini, R., Dozio, L.: A framework based on physics-informed neural networks and extreme learning for the analysis of composite structures. Comp. Struct. 265, 106761 (2022)

20. Sina Amini, Niaiki, Ehsan, Haghhiat, Trevor, Campbell, Anoush, Poursartip, and Reza, Vaziri: Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture. Computer Methods Appl. Mech. Eng. 384:113959 (2021)

21. Wessels, H., Weißenfels, C., Wriggers, P.: The neural particle method—an updated Lagrangian physics informed neural network for computational fluid dynamics. Comput. Methods Appl. Mech. Eng. 368, 113127 (2020)

22. Bing. Yu., et al.: The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Com. Math. Statistics 6(1), 1–12 (2018)

23. Yulei, Liao and Pingbing, Ming: Deep nitsche method: Deep Ritz method with essential boundary conditions. arXiv preprint arXiv:1912.01309 (2019)

24. Larry J Segerlind. Applied Finite Element Analysis. (1984)

25. Junuthula Narasimha, Reddy: An Introduction to Nonlinear Finite Element Analysis, Second Edition: With Applications to Heat Transfer, Fluid Mechanics, and Solid Mechanics. OUP Oxford (2014)

26. Esteban, Samaniego, Cosmin, Anitescu, Somdatta, Goswami, Vien Minh, Nguyen-Thanh, Hongwei, Guo, Khader, Hamdia, X Zhuang, and T Rabczuk: An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementations and applications. Computer Methods in Applied Mechanics and Engineering, 386, 113790 (2020)

27. Vien Minh, Nguyen-Thanh, Xiaoying, Zhuang, and Timon Rabczuk: A deep energy method for deep deformation hyperelasticity. European J. Mech.-A/Solids, 80:103874 (2020)

28. Jan N, Fuhg, and Nikolaos Bouklas: The mixed deep energy method for resolving concentration features in fine strain hyperelasticity. J. Comp. Phy. 451:110839 (2022)

29. Diab W, Abueidda, Seid, Korisc, Erman, Guleryuz, and Nahil A, Sobh: Enhanced physics-informed neural networks for hyperelasticity. arXiv preprint arXiv:2205.14148 (2022a)

30. Shokri, Ali, Harandi, Ahmad, Moineddin, Bai-Xiang, Xu, and Stefanie Reese: A mixed formulation for physics-informed neural networks as a potential solver for engineering problems in heterogeneous domains: comparison with finite element method. arXiv preprint arXiv:2206.13103 (2022)

31. Hunter T, Kollmann, Diab W, Abueidda, Seid, Korisc, Erman, Guleryuz, and Nahil A, Sobh: Deep learning for topology optimization of 2d metamaterials. Materials & Design, 196:109098 (2020)

32. Diab W, Abueidda, Seid, Korisc, and Nahil A, Sobh: Topology optimization of 2d structures with nonlinearities using deep learning. Computers & Struct. 237:106283 (2020)

33. Sosnovik, I., Oseledets, I.: Neural networks for topology optimization. Russ. J. Numer. Anal. Math. Model. 34(4), 215–223 (2019)

34. Saurabh, Banga, Harsh, Gehani, Sanket, Bhilare, Sagar, Patel, and Levent, Kara: 3d topology optimization using convolutional neural networks. arXiv preprint arXiv:1808.07440 (2018)

35. Zeyu Zhang, Yu., Li, W.Z., Chen, X., Yao, W., Zhao, Y.: Tonr: An exploration for a novel way combining neural network with topology optimization. Comput. Methods Appl. Mech. Eng. 386, 114083 (2021)

36. Stephan, Hoyer, Jascha, Soh-Dickstein, and Sam Greydanus: Neural reparameterization improves structural optimization. arXiv preprint arXiv:1909.04240 (2019)

37. Chandrasekhar, A., Suresh, K.: Tounn: topology optimization using neural networks. Struct. Multidiscip. Optim. 63(3), 1135–1149 (2021)

38. Zehnder, J., Li, Y., Coros, S., Thomaszewski, B.: Ntopo: Mesh-free topology optimization using implicit neural representations. Adv. Neural. Inf. Process. Syst. 34, 10368–10381 (2021)

39. Pattanayak, S.: John S Pattanayak, and Suresh John. Pro Deep Learning with Tensorflow. Springer (2017)

40. Ciyou, Zhu, Richard H, Byrd, Peihuang, Lu, and Jorge, Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on Mathematical Software (TOMS), 23(4):550–560 (1997)

41. Joshua M. Long: Random Fourier Features Pytorch. GitHub. Note: https://github.com/jmclong/random-fourier-features-pytorch (2021)

42. Bing, Yu., et al.: The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Computers & Struct. 237:106283 (2020)

43. Charnes, Esteban, Samaniego, Cosmin, Anitescu, Somdatta, Goswami, Vien Minh, Nguyen-Thanh, Hongwei, Guo, Khader, Hamdia, X Zhuang, and T Rabczuk: An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementations and applications. Computer Methods in Applied Mechanics and Engineering, 386, 113790 (2020)

44. Larry J Segerlind. Applied Finite Element Analysis. (1984)

45. Junuthula Narasimha, Reddy: An Introduction to Nonlinear Finite Element Analysis, Second Edition: With Applications to Heat Transfer, Fluid Mechanics, and Solid Mechanics. OUP Oxford (2014)

46. Esteban, Samaniego, Cosmin, Anitescu, Somdatta, Goswami, Vien Minh, Nguyen-Thanh, Hongwei, Guo, Khader, Hamdia, X Zhuang, and T Rabczuk: An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementations and applications. Computer Methods in Applied Mechanics and Engineering, 386, 113790 (2020)

47. Rozvany, G.I.N.: A critical review of established methods of structural topology optimization. Struct. Multidiscip. Optim. 30(4), 215–223 (2004)
An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., (2019). URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

52. Chandrasekhar, A., Sridhara, S., Suresh, K.: Auto: a framework for automatic differentiation in topology optimization. Struct. Multidiscip. Optim. 64(6), 4355–4365 (2021)

53. Topology optimization codes written in python. https://www.topopt.mek.dtu.dk/Apps-and-software/Topology-optimization-codes-written-in-Python. Accessed: (2022)-06-25

54. Erik, Andreassen, Anders Clausen, Mattias Schevenels, Boyan S Lazarov, and Ole Sigmund: Efficient topology optimization in matlab using 88 lines of code. Structural and Multidisciplinary Optimization, 43 (1):1–16 (2011)

55. SIMULIA. Abaqus (2020)

56. MATLAB. version R2021a. The MathWorks Inc., Natick, Massachusetts (2021)

57. Jeremy Yu, Lu Lu, Xuhui Meng, and George Em. Karniadakis: Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. Comp. Methods Appl. Mech. Eng. 393:114823 (2022)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.