Iterative Data Curation for Machine Learning-Based Inverse Design of Active Composite Plates for Four-Dimensional Printing

Teerapong Poltue, Chao Zhang, Frédéric Demoly, Kun Zhou, H. Jerry Qi

ADVANCED INTELLIGENT SYSTEMS
Open Access
www.advintellsyst.com

Iterative Data Curation for Machine Learning-Based Inverse Design of Active Composite Plates for Four-Dimensional Printing

*Teerapong Poltue, Chao Zhang, Frédéric Demoly, Kun Zhou, and H. Jerry Qi**

Active composite (AC) plates, composed of active and passive materials, can undergo complex shape transformations when stimulated. Leveraging 4D printing—which combines additive manufacturing with stimuli-responsive materials—digitally encoded design patterns offer flexibility in shape morphing. However, performing inverse design, i.e., determining the pattern to achieve a desired shape, remains challenging due to the vast design space. Recently, machine learning (ML) has been applied to inverse design tasks with promising results. Nevertheless, these approaches require large datasets, and even then, inverse design remains difficult, often demanding multiple strategies and trials to obtain optimal results. To address these challenges, this work introduces an iterative data curation strategy combined with transfer learning. This method ensures that newly curated data is nonredundant and distinct from existing datasets, reducing the required training data by a factor of eight while maintaining performance. Additionally, ML models are integrated with a genetic algorithm (ML-GA) to further fine-tune the generated design patterns. The results show that ML-GA enhances accuracy in achieving the desired shape while reducing computational effort. This framework offers an efficient and scalable approach for inverse design, reducing data needs and improving performance, making it a valuable tool for AC plate design and 4D printing.

1. Introduction

From self-folding origami to biomimetic soft robotics,[1–3] active composites (ACs) are widely used to design morphing structures, which unlock new possibilities in engineering. These composites, composed of active (responsive to stimuli) and passive (nonresponsive to stimuli) materials, exhibit controlled shape transformations upon exposure to stimuli such as heat,[4–7] light,[8,9] water,[2,10] or magnetic fields.[11–13] The advent of multimaterial 3D or 4D printing, which enables the placement of materials of distinct properties in voxels (or 3D pixels), further enables preprogrammed complex shape transformations and offers extensive fabrication versatility.[14–16] However, fully harnessing this fabrication flexibility to its full extent is still challenging. In general, a bimaterial structure with a certain property mismatch can be actuated to deform. As schematically shown in **Figure 1**a, the shape transformation of an AC plate after stimulation, the so-called actuated shape, is highly dependent on the spatial arrangement of active and passive materials (i.e., the design pattern). Due to the combinatorial nature of material distribution, such systems have a high-dimensional design space that scales exponentially with the number of voxels in the structure, making it difficult to find the design pattern for a desired actuated shape through trial and error.

Inverse design, which involves identifying a structural arrangement to produce the desired performance outcomes, constitutes an essential research topic in engineering. Previously,

T. Poltue, H. J. Qi
The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
E-mail: qih@me.gatech.edu

C. Zhang
School of Computational Science & Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202500916.

DOI: 10.1002/aisy.202500916

F. Demoly
ICB UMR 6303 CNRS
Belfort-Montbeliard University of Technology
UTBM
90010 Belfort, France

F. Demoly
Institut Universitaire de France (IUF)
75231 Paris, Frances

K. Zhou
Singapore Centre for 3D Printing
School of Mechanical and Aerospace Engineering
Nanyang Technological University
50 Nanyang Avenue, Singapore 639798, Singapore

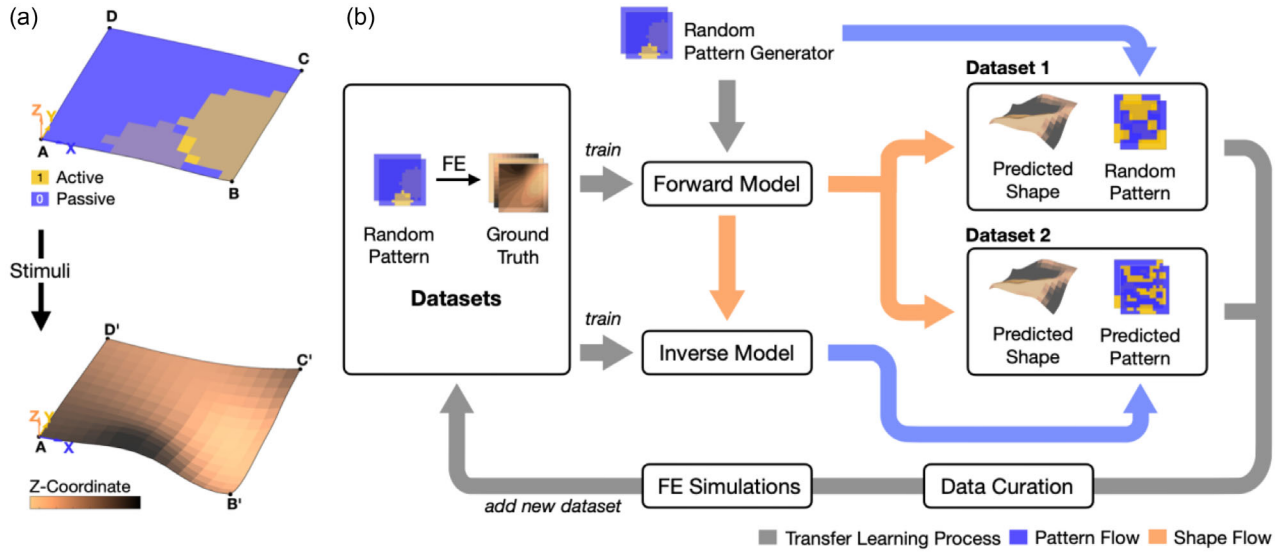

Figure 1. Overview of AC plates and ML-based forward prediction and inverse design. a) Example of an actuated AC plate after being stimulated due to the mismatch of thermal expansion coefficients of active and passive materials. b) Concept of ML-based design process: dataset generation via FE simulations, shape-change prediction using forward ML-model, design pattern generation through inverse ML-model, and data curation technique.

this challenge is typically addressed through various optimization techniques. For example, topology optimization (TO) is a versatile method that has been widely utilized to determine the optimal material distribution in a design domain for the maximization or minimization of a certain objective function (for example, but not limited to, stiffness or compliance), while fulfilling constraints such as weight or manufacturability. TO works through iterative computational approaches, which are usually coupled with the finite element (FE) method to evaluate performance, and does not require a dataset of precomputed designs and responses. The method has been successfully applied in the design of many complex structures.[17–22] However, while TO excels at structural inverse design problems, its direct application to the inverse problem of AC, which involves determining material layouts to achieve specific target shapes, is significantly more challenging. This is largely due to the highly nonlinear and non-unique solution between design pattern and deformation, in which various different patterns might form very similar deformed shapes, leading to solution instability. Apart from TO, researchers have tried reduced-order models to enhance efficiency in several inverse design problems,[23,24] but their application to AC plate deformation design is generally limited.

In the past several years, data-driven approaches, especially machine learning (ML), have become a viable complementary or alternative method for inverse design. ML models can learn intricate or nonlinear relationships between design variables and responses from extensive datasets created via simulations (such as FE) or experiments. After being trained, they are capable of making quick predictions. ML has been shown to achieve considerable success in designing structures that are optimized for particular properties, including maximizing effective modulus, controlling relative density, or improving energy absorption capabilities.[25–27] However, applying ML to the inverse design of AC structures, particularly for targeted shapes, presents special challenges. The difficulty arises from the high dimensionality of both the possible design patterns and the actuated shapes that are produced. For example, an AC plate composed of a modest number of voxels, e.g., $15 \times 15 \times 2$ used in our previous work,[28] possesses an extremely large design space, 2.9×10^{135} ($=2^{450}$). Consequently, the process of comprehending the relationship to the vast set of possible actuated geometries requires significant data. Our recent research of inverse design for AC plates using ML highlighted this need,[28] which required an initial dataset from 56 250 FE simulations, subsequently increased to 900 000 sets through data augmentation. Despite these advances, achieving reliable and computationally efficient inverse design solutions for complex shape deformations in AC plates, especially in large design spaces, remains a significant challenge.

In our previous study,[28] a promising inverse design approach was developed to achieve targeted shapes in AC plates. The approach utilized an ML model, specifically a residual network (ResNet), for forward shape prediction and integrated it with an evolutionary algorithm (EA) and a gradient descent (GD) method for inverse design. Although this previous work demonstrated a promising inverse design strategy for AC design, it faces significant challenges, including requiring a large number of FE simulations, using data augmentation to further expand data sets, converting boundary conditions from a simple support one to one similar to a clamped support, comparing two designs (from EA and GD), and employing subdomain for further improvement. We hypothesize that the reason for these challenges is mainly due to the extraordinarily large number of the design space (or possible patterns), which has threefold effects. First, even though we used 56 250 random patterns in FE simulations to create training datasets, many of them may not be significantly different, which may reduce the efficiency of these datasets. This may also leave many unseen patterns

(i.e., the randomly generated patterns may not cover the entire design space). Second, because of this, the forward prediction model still could fail to predict some actuated shapes. The fact that the predictive ML model cannot predict many shapes could further worsen the performance of the inverse design. Lastly, during the inverse design, both GD and EA could not be able to sufficiently sample the design space.

In this work, we propose two key strategies to address the aforementioned challenges. First, we implement a data curation technique to enhance the efficiency of dataset generation. Instead of relying solely on random patterns, we evaluate whether generated patterns represent unseen or seen shapes, prioritizing the inclusion of unseen designs for FE simulations and subsequent training. This is achieved by iteratively training (Figure 1b) two separate ML models: a forward ML model for rapid actuated shape prediction and an inverse ML model for pattern prediction. In each iteration, a large number of randomly generated patterns are quickly assessed by the forward ML model to predict their actuated shapes (forming dataset 1). These predicted shapes are then fed to the inverse ML model to generate corresponding patterns (forming dataset 2). We utilize the center of mass (CoM), average midsurface coordinates over the plate, to evaluate the distinctiveness of actuated shapes from these random patterns compared to previously encountered shapes, ensuring that only unseen patterns proceed to costly FE simulations for high-quality data generation. This data curation aims to significantly reduce the number of required ground-truth FE simulations, thereby lowering computational costs. Second, on the inverse design, we introduce an integrated approach called ML-GA, which combines our trained forward and inverse ML models with a genetic algorithm (GA). This strategy begins with leveraging our trained inverse ML model to propose a highly probable pattern as an initial starting point for optimization. Subsequently, the GA, in conjunction with the forward ML model, refines this design pattern. This refined inverse design strategy outperforms previous methods by requiring fewer simulations and less computational time, enabling the exploration of more complex and larger design spaces for 4D-printed AC plates and expanding their potential applications in shape-morphing structures.

It is important to note that the scope of this work is the development and validation of the computational design framework. While the resulting designs are intended for fabrication via 4D printing, the physical implementation and experimental validation of the printed structures are considered vital next steps that are beyond the scope of this current study.

2. Results and Discussion

2.1. Physical Problem and Dataset

Following our previous work,[28] as shown in Figure 1a, an active plate consists of an $N_x \times N_y \times N_z$ voxel grid, where N_x and N_y represent the number of voxels in the in-plane x- and y-direction and N_z represents the number of layers of voxels (two in this work). We use two materials, one with positive thermal expansion (referred to as active material) to represent the general stimuli-responsive materials and one with zero thermal expansion (referred to as passive material). It is noted that such choices of material properties would not lose the generality of the proposed approach, which can be modified if other stimuli-responsive materials are used. This generalized material model is a common and accepted approach in foundational computational design and TO studies, as it allows the framework to remain material-agnostic and places the focus on developing the design methodology itself. The two materials are randomly assigned into voxels, which are digitally encoded as a binary material assignment tensor (referred to as a design pattern throughout the article), $\mathbf{B}$, where "1" indicates active material and "0" represents passive material. When exposed to a stimulus, the active material expands while the passive material does not respond, causing the plate to deform. The resulting shape is captured by the coordinates (x, y, z) of voxel mesh points sampled on the midsurface, forming a tensor denoted as $\mathbf{X}$. In this article, we are working with a voxel grid of size $15 \times 15 \times 2$, where each voxel can take a value of either "0" or "1." Therefore, the total number of possible design configurations is $2^{15\times15\times2} \approx 2.9 \times 10^{135}$.

The FE model for the active plate is designed to generate the datasets (see Section 4.1). To simulate realistic behavior, we introduce a simple support boundary condition that allows the plate to deform freely. Specifically, our FE simulations implement the following boundary conditions

$$x_A = y_A = z_A = 0 \tag{1}$$

$$z_c = 0 \tag{2}$$

and constraints

$$x_C = y_C \tag{3}$$

$$z_B = z_D \tag{4}$$

where A, B, C, and D are vertices located at the corners of the plate's midsurface. As illustrated in Figure 1a, A is positioned diagonally opposite to C, and B is diagonally opposite to D. From Equation (1), node A is fixed to prevent any translation of the plate. Equation (2) and the other two constraints in Equation (3) and (4) eliminate rotation about the z-axis, as well as about the $(-1,1,0)$ and $(1,1,0)$ axes. In this work, we directly use the simulation results from such boundary conditions, without converting them into a clamped-like boundary condition.

2.2. Iterative Data Curation for Transfer Learning

As discussed before, our assumption about the reasons for the challenges in ML-based design of AC plates stemmed from the fact that the randomly generated patterns may not cover the entire possible design space, or fully randomly generated patterns may provide many similar actuated shapes (i.e., noninformative data). Since conducting FE simulations is expensive (about 1 min for each simulation), it is important to have a rapid approach to predict the actuated shape based on a given pattern. Here, we use a forward ML model to carry out this task. Simultaneously, we employ an inverse ML model, which predicts the design pattern from the actuated shape, for both transfer learning and inverse design. In other words, the inverse model

is used to generate informative patterns that support transfer learning and ultimately to provide the design pattern corresponding to a given shape. In addition, we use an iterative training approach and employ transfer learning to add the unseen datasets over iterations.[26,29] Figure 1b schematically illustrates a simplified version of the approach, while **Figure 2**a presents the detailed flowchart.

As illustrated in Figure 2a, the framework begins by generating an initial 1000 random patterns from which the corresponding actuated shapes are computed using FE simulations (see Section 4.2). These patterns consist of 25% of each of four pattern types: random circles, random rectangles, percolation patterns, and fully random patterns. Further details on pattern generation are given in Supporting Information. This process yields pairs of design patterns and the midsurface coordinates of the actuated shapes ($\mathbf{B_0}$, $\mathbf{X_0}$), which are used for the 0-th iteration of training of both forward and inverse ML models. Apparently, due to the vast design space, these initial datasets are insufficient to capture the majority of possible design configurations. To address this, we employ an iterative data curation technique, adding 200 previously unseen datasets in each iteration, until the validation loss converges.

During the i-th iteration, 2000 random patterns ($\mathbf{B_i}$) are first generated, consisting of 25% of each of the above-mentioned four patterns. The forward model then predicts the corresponding actuated shapes ($\mathbf{X_i}$) for these patterns. Since both the forward and inverse models are trained on the same datasets, it is necessary to consider the inverse model's performance when generating new datasets. To do so, the predicted actuated shapes ($\mathbf{X_i}$) are fed into the inverse ML model to obtain the predicted design patterns ($\hat{\mathbf{B}}_i$). This results in two datasets: 1) the random pair dataset, consisting of random design patterns and their corresponding predicted actuated shapes ($\mathbf{B_i}$, $\mathbf{X_i}$), and 2) the inverse pair dataset, consisting of inverse design patterns and their corresponding predicted actuated shapes ($\hat{\mathbf{B}}_i$, $\mathbf{X_i}$). It should be noted that the corresponding actuated shapes in both sets are identical, as the inverse design pattern in (2) is directly paired with the actuated shapes from the random pattern (as shown by the orange arrows in Figure 1a and 2a). This is because passing the obtained inverse pattern in (2) once again to the forward model and then pairing it with the new predicted shape may introduce cascading errors.

To evaluate the quality of the data, we calculate the CoM of each actuated shape by averaging the midsurface coordinates over the plate. The CoMs are then projected onto the plane defined by $x = y$ (i.e., the diagonal plane where $x = y$ regardless of z-values; referred to as the x–y plane). Here, we define unseen data as those whose CoMs do not fall within a 20% voxel radius in the x–y plane of existing data from previous iterations. As illustrated in Figure 2b,c, dataset pairs whose CoMs overlap with existing (seen) data ($\mathbf{B}_{0\text{ to }(i-1)}$, $\mathbf{X}_{0\text{ to }(i-1)}$) are removed, resulting in the so-called unseen datasets. From the unseen datasets, 100 datasets are selected from the random pair datasets (1) and 100 from the inverse pair dataset (2) (200 data in total) using the Latin hypercube sampling (LHS) technique.[30] These 200 patterns are then used for FE simulations, and the simulated deformed shape and the corresponding patterns are added to the training set for the next iteration. These processes are repeated until the stopping criterion is met.

The stopping criterion for the transfer learning process is defined as the point where the average validation loss stabilizes over iterations. In other words, the process continues until adding new data no longer significantly improves the model's performance.

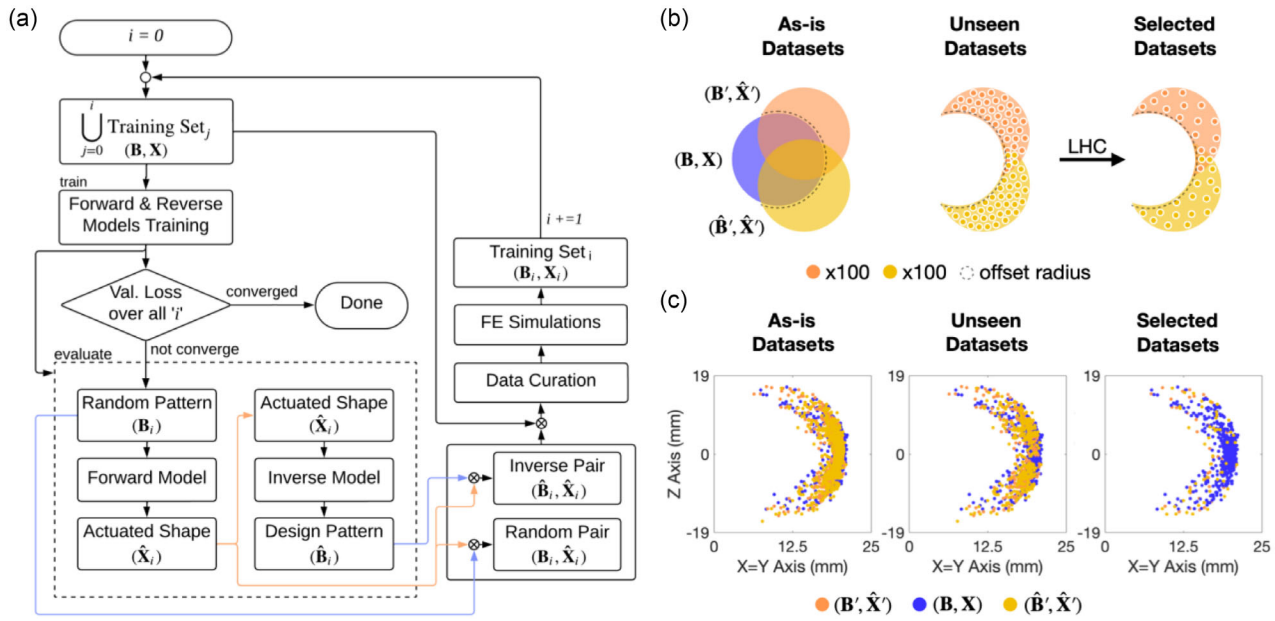

Figure 2. Iterative data curation technique for transfer learning. a) Transfer learning workflow over iterations. Val stands for validation, and FE stands for finite element simulations. b,c) Data curation technique based on the mapping of existing datasets and newly generated datasets. (b) Venn diagrams to graphically visualize the curation technique. LHS stands for Latin hypercube sampling. (c) Example of selected (curated) datasets implemented in this study based on the mapping of the projected CoMs of existing datasets and newly generated datasets onto a plane defined by $x = y$.

2.3. Machine Learning Models for Forward Prediction and Inverse Design

The ResNet architecture is known for its effectiveness in deep convolutional neural networks (CNNs) and success in various computer vision tasks like image classification.[31] Therefore, ResNet is chosen as the forward and inverse ML models.

In this work, both the forward and inverse models share the same ResNet architecture, which consists of 24 residual blocks, each containing 2 convolutional layers, as shown in **Figure 3**. In each residual block, the first convolutional layer is followed by batch normalization (BN) and an activation function, while the second layer is followed by BN only. A key feature of the ResNet architecture is the inclusion of skip connections, which allow the block input to bypass the main layers, providing a more direct path through the network. These skip connections facilitate the flow of information between shallow and deep layers, helping mitigate issues like vanishing gradients and performance degradation that are common in deep CNNs.[31]

For the forward prediction model, the input design pattern (**B**) consists of two channels representing the material layers in the thickness (z-) direction, and the output actuated shape (**X**) has three channels corresponding to the x-, y-, and z-coordinates of the midsurface. The forward model predicts continuous values; therefore, we use LeakyReLU as an activation function. The inverse model predicts binary design patterns from a targeted actuated shape; consequently, we use the sigmoid activation function. For both models, the input layer and the regression layer function as channel expansion and channel reduction layers, respectively, to match the desired number of channels.

2.4. Inverse Design Strategy

Although the inverse model can directly predict a design pattern from a targeted actuated shape, it does not fully address the complexity of the design process, especially in the one-to-many problem.[32] In evaluation (inference) mode, the inverse model provides a single solution for each targeted shape. However, this approach can be too rigid, as the model might only find one pattern, which may not be optimal for achieving the desired actuated shape across a broader design space. Additionally, while multiple forward passes could be executed, the inference mode disables certain layers (such as dropout) that introduce randomness during training, and BN uses fixed statistics instead of batch statistics. This ensures that the inverse model's outputs are deterministic;[33] in other words, it limits the diversity of possible patterns the inverse model can generate. To address this

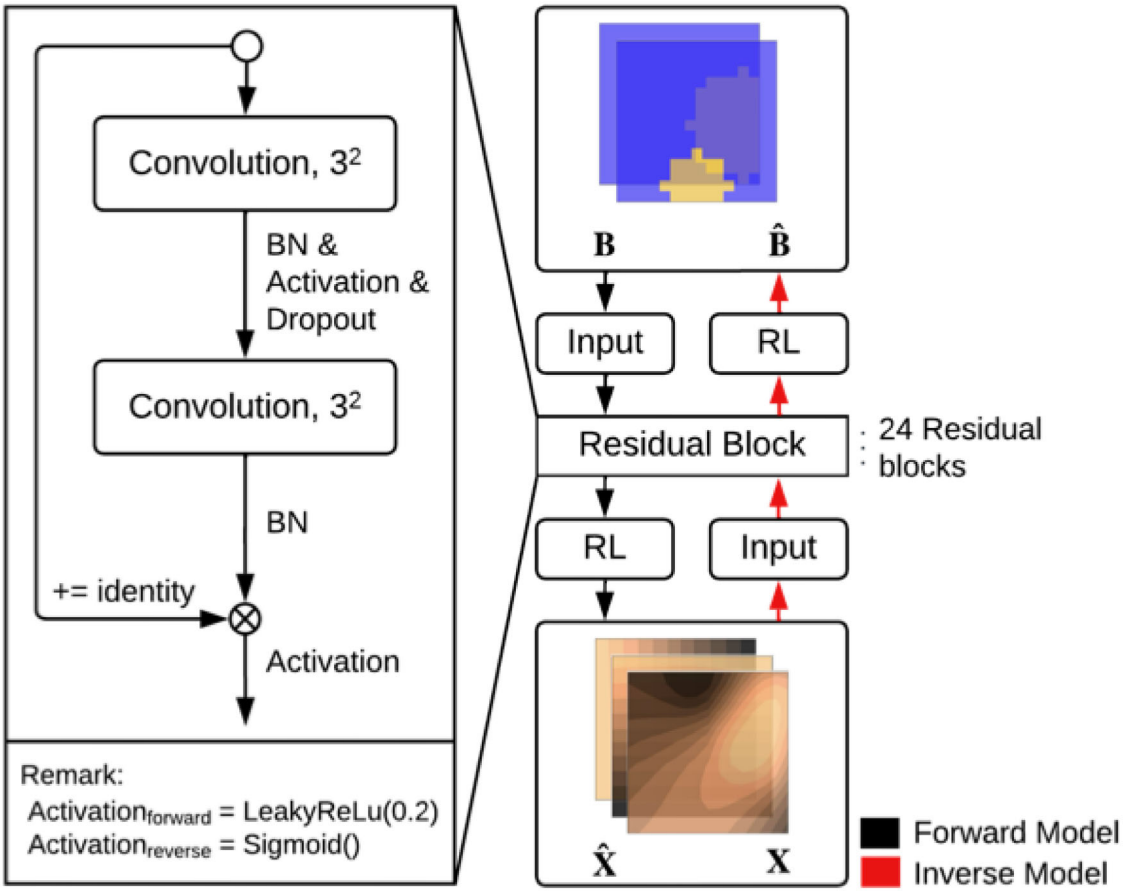

Figure 3. Overview of ResNet based on forward prediction and inverse design and their residual blocks for 4D-printed AC plates. BN stands for batch normalization, and RL stands for regression layer.

one-to-many challenge, one could employ iterative, gradient-based techniques that refine a design starting from an initial guess. For instance, backpropagation can be applied to a trained forward model to iteratively update the input design pattern to minimize the error with a target shape, a strategy seen in frameworks like the "predictor-designer" model by Chen and Gu.[34] Similarly, iterative Jacobian-based methods are effective for sequential tasks like robotic path following, as shown by Fang et al.[35] However, these methods can be sensitive to the initial guess and may converge to local minima in a vast design space like ours.

While more advanced metrics like Chamfer distance are widely recognized for their effectiveness in comparing unstructured point clouds and achieving better convergence in 3D geometry optimization, our study utilizes mean squared error (MSE) as the fitness function within the GA. The choice of MSE is a deliberate one, driven by the structured nature of our dataset. The FE simulations produce actuated shapes with a consistent, aligned coordinate system, ensuring a direct correspondence between the mesh points of the predicted and target shapes. This allows MSE to function as a computationally efficient and accurate metric for evaluating fitness. The success of this approach is further attributed to the ML-GA framework, where the GA's ability to navigate nonconvex and discontinuous search spaces mitigates the potential for local minima that might arise from using a simpler loss function in a gradient-based method. However, as the field of 4D printing advances toward more complex, nonaligned geometries, future work may benefit from exploring alternative loss functions such as the Chamfer distance. This would further enhance the robustness and generalizability of the inverse design framework for broader applications.

Our approach, therefore, uses the trained inverse model not as the final solver, but as an efficient generator of a high-quality initial guess. This guess then seeds a more robust global optimization method, allowing for broader exploration of the solution space. Therefore, to explore a larger design space and account for more complex design requirements, we integrate the ML models with a GA (ML-GA), which allows for a more flexible and optimal solution.

Typically, a GA-encoded sequence of each individual is represented using binary values and referred to as a gene. The GA iteratively evaluates and evolves the gene over generations. As shown in **Figure 4**a, in each generation, a population of 200 individuals is evaluated simultaneously using the forward ML model. The predicted shapes of the population are then compared to the targeted shape using the MSE as the fitness score function. While more complex metrics such as the Chamfer distance are effective for comparing arbitrary point clouds, we utilize pointwise MSE because our FE simulation framework produces actuated shapes in a consistent, aligned coordinate system. This provides a direct correspondence between the mesh points of the predicted and target shapes, making MSE a computationally efficient and accurate metric for evaluating fitness within the iterative GA loop. Based on the fitness scores, the GA creates an offspring population for the next generation, comprising individuals from the current population, specifically the elite (5%) and mating pool (95%). The top 5% individuals in the population (elites) with the lowest MSE scores are automatically retained for the next generation. The remaining 95% of individuals in the mating pool are paired as parents using tournament selection before crossover and mutation operations.

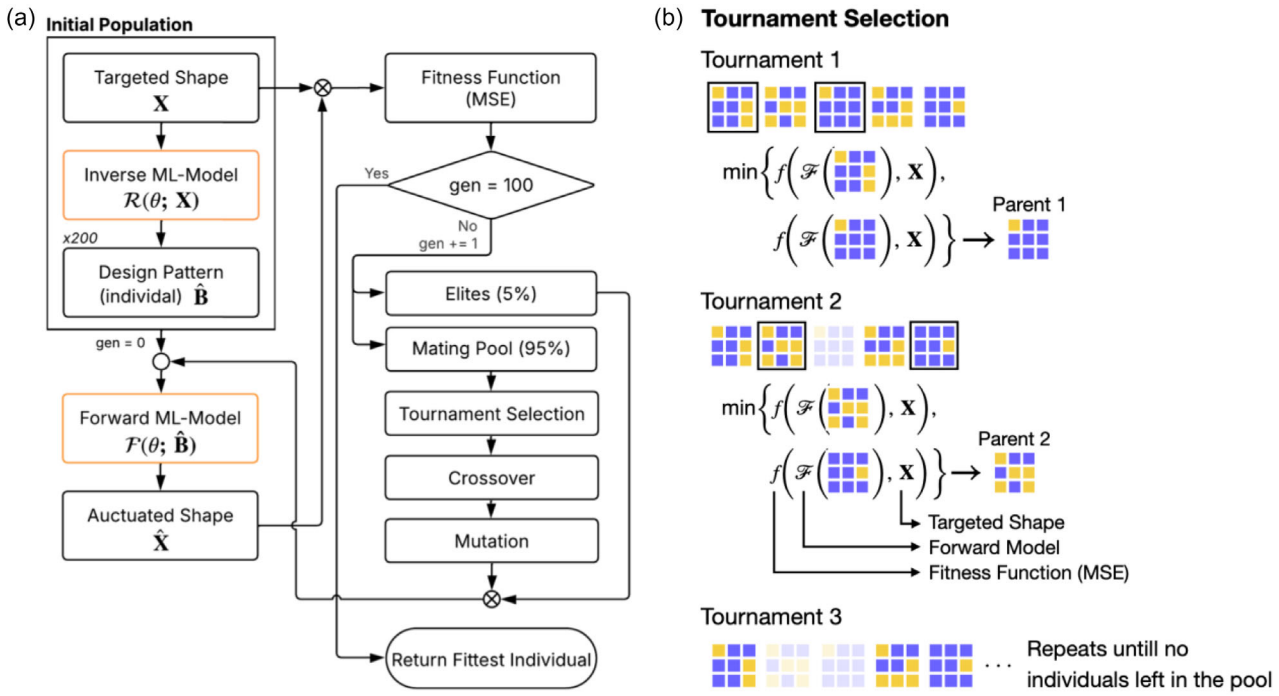

Figure 4. An integrative method of the forward and inverse ML models with a GA (ML-GA) and graphical representation of tournament selection. a) Flowchart illustrating the inverse design process, starting with the desired actuated shape, using ML-GA. b) Graphical representation outlining the tournament selection process performed to get a pair of parents.

Figure 4b shows how a pair of parents is formed, starting by randomly selecting two individuals from the mating pool to compare their fitness scores. The fittest individual is retained, and the process is repeated once more to complete the pair (i.e., two tournaments are conducted to obtain a pair). This pairing method is repeated until all individuals in the mating pool have been paired. In addition, it must be noted that each individual is only allowed to be paired once. In other words, there is no repeated individual across all pairs. After all individuals are paired, crossover is performed on each pair of parents. Regarding the crossover operation, a section of the gene of each individual in the parents is swapped between them, resulting in children. Finally, the obtained children are mutated, where certain sequences of individuals are randomly converted from their current assignment (i.e., from "0" to "1" or vice versa). It is noteworthy that not all parents and children will undergo crossover and mutation, as the probabilities of implementing these processes are set to 0.5 and 0.05, respectively. Notably, these probabilities were chosen based on common practices in GA literatures and were found empirically to provide sufficient diversity and convergence behavior for this problem.[36,37] These operations, namely, elite selection, tournament selection, crossover, and mutation, are repeated for 100 generations. Subsequently, the most successful individual that mimics the desired shape is returned as a design pattern.

The choice of a GA over gradient-based optimizers like Adam is deliberate and motivated by the nature of our design problem. The design pattern is inherently discrete and binary (a voxel is either active material "1" or passive material "0"). GAs are naturally suited for such combinatorial optimization problems and excel at exploring nonconvex and discontinuous search spaces to avoid local optima. In contrast, gradient-based methods require a continuous and differentiable design space. Advanced TO works, such as those by Wang et al. and Han et al.,[38,39] successfully employ Adam by optimizing continuous variables like the latent codes or weights of a neural network that implicitly define the structure. Our ML-GA approach, however, directly addresses the discrete design variables. Furthermore, we mitigate the primary drawback of GAs—slow convergence—by seeding the initial population with a high-quality solution from our inverse ML model. This synergy makes the ML-GA framework a highly efficient and effective strategy for this specific class of inverse design problems.

Figure 4a also depicts the ML-GA workflow, where the initial population for the GA is generated using the inverse ML model, instead of random initialization. Since the inverse ML model only provides a single solution, it will subsequently be replicated to serve as the initial population (i.e., several individuals). It is noteworthy that in the first generation, all individuals in the population are identical, as well as their fitness scores. However, after the tournament selection, crossover, and mutation operations are implemented, a more diverse population will be observed in the subsequent generation.

2.5. Performance of the Forward ML Model

Figure 5a,b shows the validation loss of the forward and inverse ML models, respectively, for the 10th, 30th, and 50th iterations. It is important to note that training starts at the 0-th iteration with 1000 datasets, and at each iteration, 200 previously unseen data are added. The validation loss of the inverse ML model at iteration 30th was significantly reduced, as compared to iteration the 10th (from 0.1755 to 0.0149) due to the increased dataset (4000 unseen datasets were added). This loss was computed as the MSE between the predicted and targeted actuated coordinates. Note that targeted actuated coordinates were normalized prior to training (see Section 4.3), which scales the coordinates (ranging from 0 to 40) to zero mean and unit variance to balance their influence during model optimization. From iteration the 30th to the 50th, a reduction from 0.0149 to 0.0089 in loss was observed even though the same number of datasets was added compared to the period from the 10th to the 30th iteration, indicating diminishing returns in terms of performance improvement. This suggests that by the 30th iteration, the model has already achieved most of its performance gains, making additional iterations less impactful. As shown in Figure 5c, at the beginning of transfer learning, adding unseen data significantly improves the model's performance, followed by a more gradual improvement in later iterations.

Figure 5d shows the 2D histogram of the ground-truth coordinates versus the forward ML predicted coordinates from the validation set. It is worth mentioning that the example from the 30th iteration is chosen to represent the performance in this section and for the rest of the article, as it marks the point where the validation loss begins to converge over iterations. It is clearly observed that the data points are concentrated along the regression line with a slope of 1, with $R^2 > 0.99$, indicating excellent accuracy of the forward ML model. The R^2, or coefficient of determination, is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variables. A value closer to 1 indicates perfect prediction, while a value closer to 0 represents poor prediction. Interestingly, in this study, the R^2 value of the forward model in the validation set at the 30th iteration is 0.9995, which is slightly higher than the 0.9980 reported by our previous work.[28] For the ML model training, in our previous work, we used 56 250 FE simulations, which required ≈39 days on a single CPU computer and were expanded. In addition, to effectively train the model those simulations were augmented to 900 000 before training. Notably, while our previous work used 56 250 datasets (with data augmentation) for training, only 7000 datasets (without data augmentation) are used in this work (in the 30th iteration). While the higher R^2 in this study suggests excellent accuracy, it is important to note that comparing R^2 across models trained on different dataset sizes may not be entirely fair. A smaller dataset can sometimes lead to a higher R^2 because the model has fewer variations to learn from, making it easier to fit the data closely (i.e., overfitting)—though this does not always mean better generalization. In contrast, a larger dataset introduces more diversity, which can slightly lower R^2, but often leads to a more robust model and more generalized model that can handle a wider range of data. To ensure a fair comparison, Section 2.6 will evaluate the proposed model on the same inverse design problem using the same test set with Sun et al.'s work.[28]

The performance of the current model is further illustrated in the surface plot of selected two examples from the validation set,

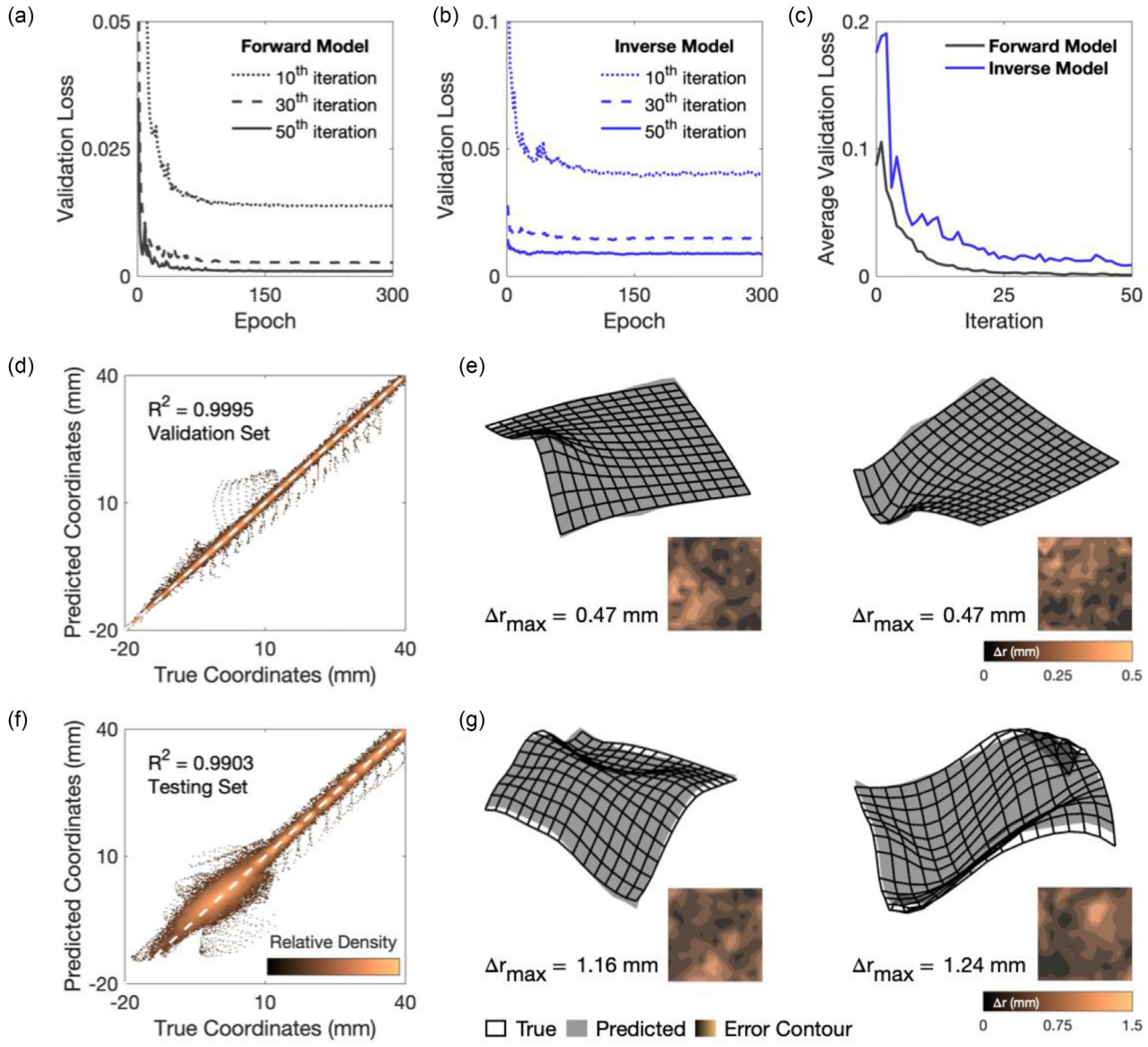

Figure 5. Overall performance of forward and inverse ML models. a,b) Examples of validation loss for (a) forward and (b) inverse models over epochs in three different training iterations. c) Performance of iterative data curation in transfer learning of forward and inverse models over iterations, in which in each iteration 200 unseen datasets are added. d,f) Density scatter plot (2D histogram) of the true coordinates and predicted coordinates for (d) forward model at 30th iteration of validation and (f) testing set. e,g) Comparison of ground-truth (black grid) and forward ML-predicted shape (gray surface) from design pattern along with error contour, randomly picked from the (e) validation and (g) testing set.

as shown in Figure 5e where the ground-truth shape (white grid) is plotted against the predicted surface (color contour). The contour represents the local absolute distance error (Δr), referred to as the error for simplicity throughout the article, between the ground truth and predicted shape

$$\Delta r = \sqrt{(x-\hat{x})^2 + (y-\hat{y})^2 + (z-\hat{z})^2} \tag{5}$$

where x, y, and z are FE-derived midsurface coordinates and $\hat{x}$, $\hat{y}$, and $\hat{z}$ are predicted coordinates. As shown in Figure 5e, the surface plots from the validation set of the forward ML model demonstrate exceptional performance, with an error range of less than 1 mm.

To further demonstrate the performance of the forward model on completely unseen datasets, Figure 5f illustrates the density scatter plot of the true shape versus the predicted shape for 500 testing samples. As expected, for the testing sets, the R^2 shows a slight drop (0.92%) compared to the validation set. However, the model still exhibits excellent performance, achieving $R^2 > 0.99$. Additionally, the performance on the testing set is further visualized through two surface plots in Figure 5g, which shows that the error can be as low as 1.16 mm (or 2.9%, after normalizing by the edge length of the plate, 40 mm).

2.6. Performance of the Inverse Design

Similar to the results observed in the forward ML model, **Figure 6**a also shows high correlation in the inverse ML model between the targeted coordinates and computed (FE-derived) coordinates in the validation set, with $R^2 > 0.99$, indicating excellent accuracy of the inverse model. To further test the performance of the inverse ML model, 500 unseen testing sets were utilized. Figure 6b shows the 2D histogram of the targeted coordinates and computed coordinates of design patterns from the inverse ML model in the testing set, where $R^2 > 0.99$ is shown. Notably, to accurately assess the performance of the inverse design, the obtained inverse design patterns are evaluated using FE simulation to compute the actuated shapes, rather than using the forward ML model. Using the forward ML model for evaluation would introduce cascading errors due to the inherent errors of the forward ML model itself. Therefore, the forward ML model is used solely during the inverse design process, not for performance evaluation.

As discussed in Section 2.4, a single target shape does not necessarily correspond to a unique solution. Since in the evaluation mode the ML model produces consistent outputs, ensuring deterministic behavior, an ML-GA is proposed to enhance the inverse design capability. Figure 6c shows a slight increase in R^2 of the proposed ML-GA method, improving from 0.9935 to 0.9976. However, a more notable improvement is shown in Figure 6d, where the maximum distance error distribution is presented. The ML-GA method significantly reduces the maximum distance error, as seen in the histogram where the peak increases from 90 to 160 counts at the lowest bound (indicating that more testing sets evaluated on the ML-GA outperform the inverse ML model). Additionally, the maximum error of the top 90% performers in testing sets is reduced by 40% (i.e., from 2.5 mm in the inverse ML model to 1.5 mm in the ML-GA method). These results (Figure 6c,d) demonstrate the effectiveness of the proposed ML-GA method, which significantly enhances the inverse design pattern to achieve the targeted shape.

For a fair comparison with the previous work by Sun et al.,[28] the benchmark targeted shapes from the previous study are used to evaluate the performance of the inverse design in this study. **Figure 7** illustrates the achieved shapes from both the ML inverse design (second column) and the ML-GA inverse design (third column). Both methods reaffirm that a single targeted shape does not necessarily correspond to a single unique solution. As shown in the design patterns (to the right of each surface) for the same targeted shape, the patterns are not entirely

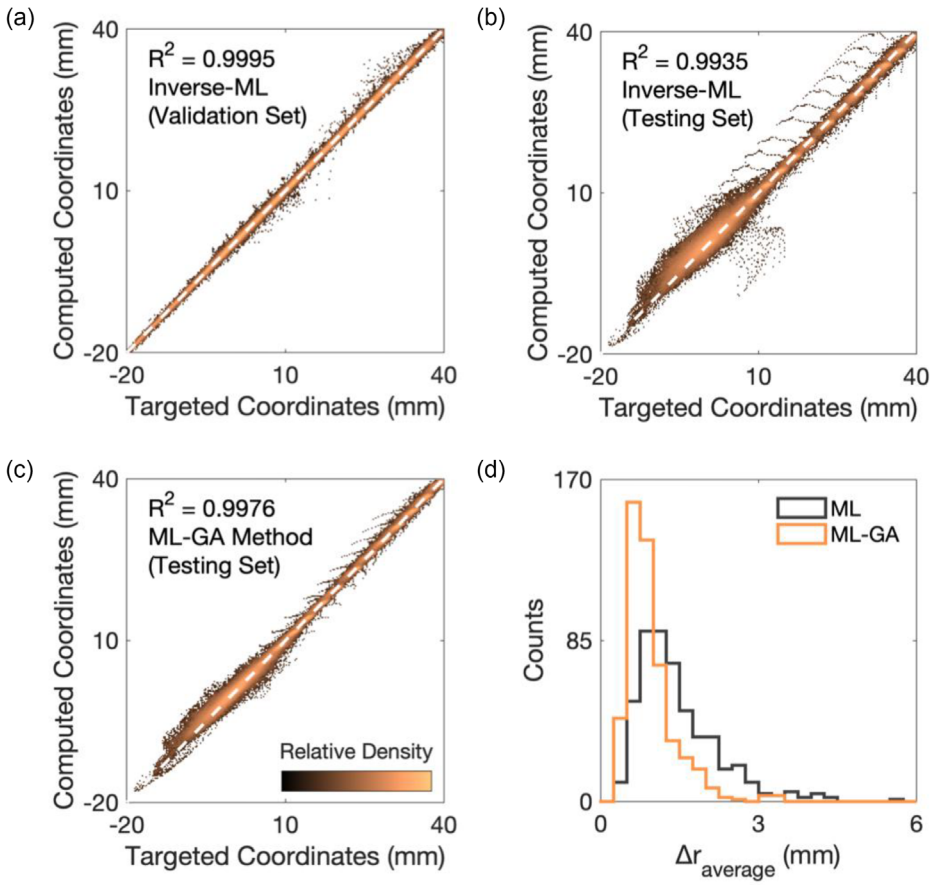

Figure 6. Inverse design performance based on the proposed strategy using the model trained at 30th iteration. Inverse ML model: a,b) 2D histogram of the targeted coordinates and computed (FE-derived) coordinates of design patterns in the (a) validation set and (b) testing set. c) 2D histogram of the targeted coordinates and computed coordinates of design patterns from the proposed ML-GA method in 500 testing sets. d) Comparison of average error distribution across 500 testing sets of the inverse ML model and the ML-GA method.

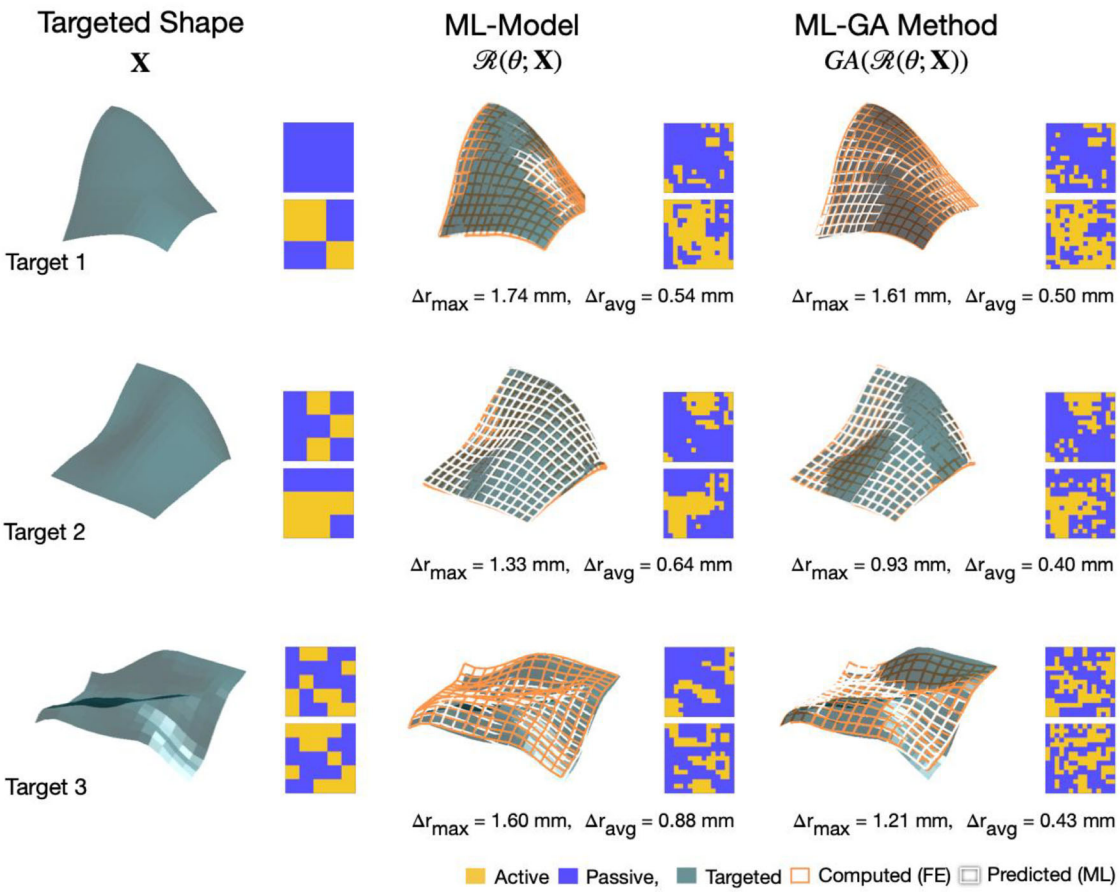

Figure 7. A performance of inverse design from 30th iteration is presented using benchmark targeted shapes from previous work by Sun et al.[28] The first, second, and third columns display the targeted shapes, the inverse design performance of the ML model, and the inverse design performance of the ML-GA method along with their corresponding design patterns, respectively. The targeted shape is shown as a dark-gray surface, while the achieved shape, predicted by the forward ML model and computed by the FE method, is represented by white and orange grids, respectively.

identical, yet certain common features emerge across inverse design schemes. For example, in the bottom layer of target 1, the active voxels align diagonally in the targeted design, a feature consistently observed in both inverse design methods (last two columns). Similarly, in the bottom layer of target 2, the active voxels form an L-like shape (rotated 90° clockwise) in the targeted design, which is also evident in every inverse design scheme.

For quantitative comparison, the error is computed and normalized by the edge length of the plate (40 mm). **Table 1** compares the maximum error and average error for each targeted shape from previous and current studies. The inverse ML model shows comparable error to the previous study, with a significant reduction (87.5%) in the number of datasets used in training. Nonetheless, some trade-offs must be pointed out. For instance, the average errors of targets 2 and 3 of the ML model in this current study are slightly higher than in the previous work. However, when the proposed ML-GA method is utilized, even with the model being trained with a much smaller dataset in this study, more accurate results are achieved. As indicated by the maximum and average error for all targeted shapes, both values are lower than in the previous study. In addition to the superior performance of the current study, it is worth mentioning the implied computational cost reduction, given that at the 30th iteration, it required reducing 87.5% of the datasets. This also means that the computational time required for data generation is significantly reduced as well (from ≈39 to ≈4.8 days in single-core computation).

Table 1. Quantitative comparison of normalized distance error, for each targeted shape in Figure 7, between the previous study by Sun et al.[28] and proposed inverse design schemes.

Target	Sun et al.		Current study			
			Inverse ML model		ML-GA method	
	Δr_{max} [mm]	Δr_{avg} [mm]	Δr_{max} [mm]	Δr_{avg} [mm]	Δr_{max} [mm]	Δr_{avg} [mm]
Target 1	1.95	0.62	1.74	0.54	1.61	0.50
Target 2	1.38	0.51	1.33	0.64	0.93	0.40
Target 3	1.56	0.60	1.60	0.88	1.21	0.43

In addition to a quantitative comparison of previous and current studies, it is important to mention the computational costs involved when the inverse design is performed. **Table 2** breaks

Table 2. Time cost comparison of each targeted shape in Figure 7, between the previous study by Sun et al.[28] and proposed inverse design schemes.

Target	Sun et al. [s]	Current study [s]	
		ML model	ML-GA method
Target 1	216	3.2	25.8
Target 2	720	3.5	30.2
Target 3	792	2.8	23.4

down the computational cost required to inverse design the targeted shapes in Figure 7. In every targeted shape, the current study achieves significantly faster inverse designs compared to the previous study, which required ≈3–13 min, whereas our new approach takes less than a minute. The key factor contributing to this time difference is the inverse design strategy used in each study. In the previous work, the inverse design process involved several steps: first, comparing the results of ML-gradient descent and ML-EA (2000 individuals required) and then using subdomain optimization to further refine the obtained patterns. In contrast, the ML-GA method in this study benefits from an initial guess provided by the inverse ML model, allowing the GA to converge with only 30 individuals. Based on the convergence test, using more than 30 individuals showed no significant difference in the final result, except for the increased calculation time. In other words, the ML-GA method starts with a more informed initial guess, leading to a more efficient search for the local optimum design.

3. Conclusion

We introduce an efficient method for predicting the actuated shape and performing the inverse design task for 4D-printed AC plates using ML and GA. The iterative data curation technique with transfer learning is proposed to reduce the number of datasets required in training the ML models, while maintaining the accuracy of the forward prediction and inverse design. The key factor in iterative data curation is the systematic comparison of new random datasets with existing datasets based on a critical feature—in this study, the centers of mass (CoMs). This ensures that they do not contain relatively identical, noninformative datasets before these unseen sets are added to the training for the next iteration. In addition, to ensure that the curated data will cover all possible regions, an LHS technique is employed to systematically choose data from unseen sets. This curation technique is applied to both forward and inverse ML models; it has been proven to effectively reduce the number of datasets used in training from 56 250 sets (before data augmentation) in previous work to only 7000 sets in this work, while showing comparable performance in inverse design of the benchmark targeted shapes. Furthermore, an ML-GA method is proposed to integrate the inverse ML model with the GA, providing an initial guess from the inverse ML model to the GA as a seed. This method has been proven to obtain superior results for the benchmark targeted shapes compared to the previous work, while the prediction time is significantly reduced from ≈3–13 min per targeted shape to less than a minute herein. This improvement is due to the reduced number of individuals required for the GA to converge to an optimal solution, thanks to the initial guess provided by the inverse ML model. Therefore, our inverse design approach paves the way for a more complex and larger design space of 4D-printed AC plates and is promising in facilitating broader applications of 4D-printed shape-morphing AC structures.

It should be noted that the validation presented in this study is computational, where the performance of the proposed ML-based framework is rigorously benchmarked against high-fidelity FE simulations. This focus is consistent with foundational studies in computational design, where the primary contribution is the development of a novel design methodology. However, we acknowledge that physical fabrication and experimental testing of the generated designs are crucial for ultimate real-world application. Therefore, the experimental validation of the shape-morphing capabilities of AC plates designed using our framework represents an important and promising direction for future research.

4. Experimental Section

FE Setup: FE analysis was performed using a commercial software package, Abaqus (version 2023, Simulia, Providence, RI), to generate the dataset to be utilized in training ML models. For the problem setup, both active and passive materials were modeled using the incompressible neo-Hookean model with the same elastic modulus. The actuated shape was triggered by the thermal expansion in the simulation. To achieve the actuated shape after the plate was actuated, a mismatch in the thermal expansion coefficients of the active and passive materials was assigned as 0.001 and 0, respectively. An increase of temperature to 50 °C was applied to the entire domain, resulting in the active material phase being subjected to a linear strain of 0.05. The plate was $40 \times 40 \times 1$ mm in size, which was meshed into $45 \times 45 \times 4 = 8100$ C3D8H (eight-node linear brick, hybrid) elements as this mesh size ensured convergence in the displacement field. The plate was divided into $N_x \times N_y \times N_z$ (i.e., $15 \times 15 \times 2$) voxels for material assignment, in which each voxel was composed of $(45/N_x) \times (45/N_y) \times (4/N_z)$ (i.e., $3 \times 3 \times 2$) meshed elements. In Abaqus, materials were assigned directly to each voxel's mesh without defining contact interfaces between adjacent voxels. Instead, material interaction occurred through a continuous mesh with shared nodes, ensuring displacement compatibility across voxels without the need for tie or contact definitions. Note that during the shape morphing, the self-contact of the plate was not considered. To manage the large number of simulations required, an Abaqus Python macro script was employed by reading the binary tensor of "1" (active) and "0" (passive) from the design pattern.

Dataset Preparation: Initially, FE generated 1000 sets of the coordinates of all nodes in a voxel on midsurface of the actuated plate as the ground truth, resulting in a tensor $\mathbf{X}$ with shape $1000 \times 3 \times 15 \times 15$, where 3 represents the coordinates in the *x*-, *y*-, and *z*-directions. These were later paired with a design pattern tensor $\mathbf{B}$ with shape $1000 \times 2 \times 15 \times 15$, where 2 represented each layer of the plate. The initial 1000 design patterns were randomly generated from four designs (250 each), including random circle, random rectangle, percolation pattern, and fully random pattern using the *randint* function in the PyTorch library, where the examples of these patterns are presented in S1, Supporting Information. Design pattern generation. Finally, the dataset was split into training and validation sets with fractions of 0.8 and 0.2, respectively.

For FE data generation, the computation is performed in parallel across ten groups, each using a single core of an Intel Core i9-10900 CPU. The initial set took ≈100 min to generate. In each subsequent iteration, 200 additional sets were added to the training data, requiring an additional 20 min of computation per iteration.

Machine Learning Models: ResNet was used for both forward prediction and inverse design of the active plate, and it consisted of input layer, 24 residual blocks (2 layers each), and a regression layer. All the convolutional layers, including the input layer, residual blocks, and regression layer, utilized a 2D kernel size of 3×3. Apart from the difference in input and output channels (i.e., 2–3 for the forward model and 3–2 for the inverse model), the primary distinction between the two models was the activation function. LeakyReLU was used for forward prediction, while sigmoid was chosen for inverse design, as the latter required a design pattern, whereas the forward model predicted continuous values. Given that the coordinate values (≈0–40) of the actuated plate were relatively much greater than those of the design pattern (0 or 1), *StandardScaler* was used to normalize the coordinates to have a mean of 0 and a standard deviation of 1. The scaling followed the equation $x' = (x - \mu)/\sigma$, where x' is the scaled data, x is the original data, μ is the mean of the original data, and σ is the standard deviation of the original data. Regarding hyperparameters, the learning rate was initially set to 0.001 and multiplied by a factor of $1/\sqrt{2}$ every 12 epochs. The batch size was set to 512, and the Adam optimizer was used for updating parameters over 300 epochs. The model was trained on a single NVIDIA Tesla V100 GPU.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

The support of AFOSR grant (A9550-20-1-0306; Dr. B.-L. "Les" Lee, Program Manager) was gratefully acknowledged.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

4D printing, active composites, data curation, machine learning

Received: August 12, 2025
Revised: September 25, 2025
Published online:

[1] B. C. Jian, F. Demoly, Y. C. Zhang, H. J. Qi, J. C. André, S. Gomes, *Engineering* **2022**, *12*, 70.
[2] A. Sydney Gladman, E. A. Matsumoto, R. G. Nuzzo, L. Mahadevan, J. A. Lewis, *Nat. Mater.* **2016**, *15*, 413.
[3] A. Zolfagharian, M. Denk, M. Bodaghi, A. Z. Kouzani, A. Kaynak, *AcMSS* **2020**, *33*, 418.
[4] Y. Mao, K. Yu, M. S. Isakov, J. Wu, M. L. Dunn, H. Jerry Qi, *Sci. Rep.* **2015**, *5*, 13616.
[5] D. J. Roach, X. Kuang, C. Yuan, K. Chen, H. J. Qi, *Smart Mater. Struct.* **2018**, *27*, 125011.
[6] X. Peng, S. Wu, X. Sun, L. Yue, S. M. Montgomery, F. Demoly, K. Zhou, R. R. Zhao, H. J. Qi, *Adv. Mater.* **2022**, *34*, 2204890.
[7] D. J. Roach, X. Sun, X. Peng, F. Demoly, K. Zhou, H. J. Qi, *Adv. Funct. Mater.* **2022**, *32*, 2203236.
[8] A. Lendlein, H. Jiang, O. Jünger, R. Langer, *Nature* **2005**, *434*, 879.
[9] X. Mu, N. Sowan, J. A. Tumbic, C. N. Bowman, P. T. Mather, H. J. Qi, *Soft Matter* **2015**, *11*, 2673.
[10] Z. Zhao, X. Kuang, C. Yuan, H. J. Qi, D. Fang, *ACS Appl. Mater. Interfaces* **2018**, *10*, 19932.
[11] X. Kuang, S. Wu, Q. Ze, L. Yue, Y. Jin, S. M. Montgomery, F. Yang, H. J. Qi, R. Zhao, *Adv. Mater.* **2021**, *33*, 2102113.
[12] S. M. Montgomery, S. Wu, X. Kuang, C. D. Armstrong, C. Zemelka, Q. Ze, R. Zhang, R. Zhao, H. J. Qi, *Adv. Funct. Mater.* **2021**, *31*, 2005319.
[13] Q. Ze, S. Wu, J. Dai, S. Leanza, G. Ikeda, P. C. Yang, G. Iaccarino, R. R. Zhao, *Nat. Commun.* **2022**, *13*, 3118.
[14] Q. Ge, H. J. Qi, M. L. Dunn, *Appl. Phys. Lett.* **2013**, *103*, 131901.
[15] X. Kuang, D. J. Roach, J. Wu, C. M. Hamel, Z. Ding, T. Wang, M. L. Dunn, H. J. Qi, *Adv. Funct. Mater.* **2019**, *29*, 1805290.
[16] L. Yue, S. Macrae Montgomery, X. Sun, L. Yu, Y. Song, T. Nomura, M. Tanaka, H. Jerry Qi, *Nat. Commun.* **2023**, *14*, 1251.
[17] K. Maute, A. Tkachuk, J. Wu, H. Jerry Qi, Z. Ding, M. L. Dunn, *J. Mech. Des.* **2015**, *137*, 111402.
[18] M. J. Geiss, N. Boddeti, O. Weeger, K. Maute, M. L. Dunn, *J. Mech. Des.* **2019**, *141*, 051405.
[19] G. Sossou, F. Demoly, H. Belkebir, H. J. Qi, S. Gomes, G. Montavon, *Mater. Des.* **2019**, *181*, 108074.
[20] L. Wang, D. Zheng, P. Harker, A. B. Patel, C. F. Guo, X. Zhao, *Proc. Natl. Acad. Sci.* **2021**, *118*, e2021922118.
[21] M. Tanaka, S. M. Montgomery, L. Yue, Y. Wei, Y. Song, T. Nomura, H. J. Qi, *Sci. Adv.* **2023**, *9*, eade4381.
[22] D. Athinarayanarao, R. Prod'hon, D. Chamoret, H. J. Qi, M. Bodaghi, J.-C. André, F. Demoly, *NPJ Comput. Mater.* **2023**, *9*, 1.
[23] T. Cui, Y. M. Marzouk, K. E. Willcox, *Int. J. Numer. Methods Eng.* **2015**, *102*, 966.
[24] G. Sossou, F. Demoly, H. Belkebir, H. J. Qi, S. Gomes, G. Montavon, *Mater. Des.* **2019**, *181*, 108074.
[25] H. Pahlavani, K. Tsifoutis-Kazolis, M. C. Saldivar, P. Mody, J. Zhou, M. J. Mirzaali, A. A. Zadpoor, *Adv. Mater.* **2024**, *36*, 2303481.
[26] M. W. Cho, K. Ko, M. Mohammadhosseinzadeh, J. H. Kim, D. Y. Park, S. M. Park, *Mater. Horiz.* **2024**, *11*, 2615.
[27] X. Zheng, T.-T. Chen, X. Jiang, M. Naito, I. Watanabe, *Sci. Technol. Adv. Mater.* **2023**, *24*, 2157682.
[28] X. Sun, L. Yue, L. Yu, C. T. Forte, C. D. Armstrong, K. Zhou, F. Demoly, R. R. Zhao, H. J. Qi, *Nat. Commun.* **2024**, *15*, 5509.
[29] Z. Zhang, Q. Liu, D. Wu, *Mater. Des.* **2022**, *218*, 110700.
[30] M. Stein, *Technometrics* **1987**, *29*, 143.
[31] K. He, X. Zhang, S. Ren, J. Sun, presented at *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition* **2016**, https://doi.org/10.1080/00207160701303912.
[32] P. Naseri, S. V. Hum, *IEEE Trans. Antennas Propag.* **2021**, *69*, 5725.
[33] V. Papyan, Y. Romano, M. Elad, *J. Mach. Learn. Res.* **2017**, *18*, https://doi.org/10.48550/arXiv.1607.08194.
[34] C. T. Chen, G. X. Gu, *Adv. Sci.* **2020**, *7*, 2000878.
[35] G. X. Fang, Y. J. Tian, Z. X. Yang, J. M. P. Geraedts, C. C. L. Wang, *IEEE/ASME Trans. Mechatron.* **2022**, *27*, 5296.
[36] T. Back, I. Ebrary, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, New York **1996**.
[37] R. M. Losee, *Inf. Process. Manage.* **1997**, *33*, 407.
[38] X. J. Han, Y. Jiang, W. M. Wang, G. X. Fang, S. Gill, Z. Q. Zhang, S. F. Wang, J. Saito, D. Kumar, Z. X. Luo, E. Whiting, C. C. L. Wang, in *Proc. Siggraph Asia 2024 Conf. Papers* **2024**, https://doi.org/10.1145/3680528.3687661.
[39] W. Wang, Y. Hou, R. Su, W. Wang, C. C. Wang, *Adv. Intell. Discovery* **2025**, *1*, 2400015.