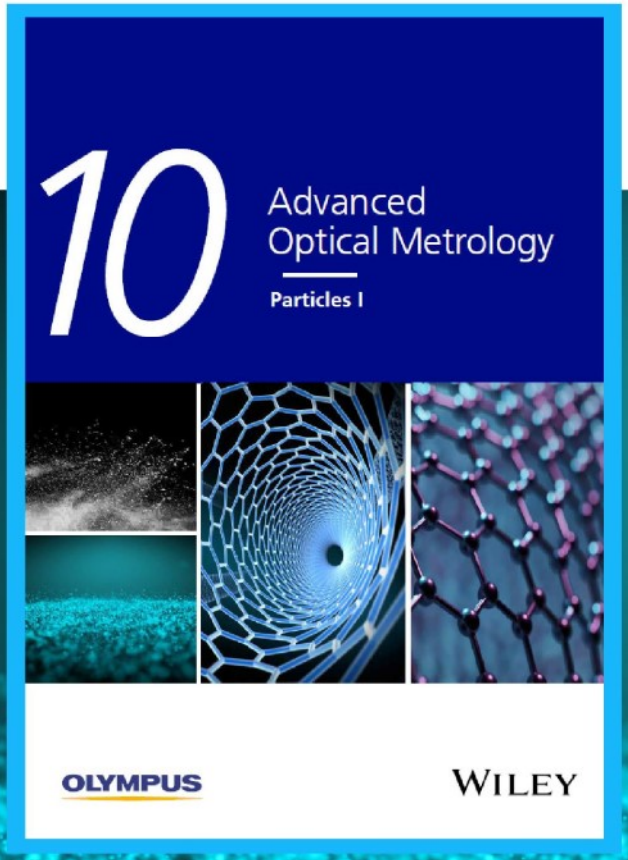
Particles I

Access the latest eBook →



Particles: Unique Properties, Uncountable Applications

Read the latest eBook and better your knowledge with highlights from the recent studies on the design and characterization of micro- and nanoparticles for different application areas.

Access Now

This eBook is sponsored by

OLYMPUS

WILEY

ADVANCED FUNCTIONAL MATERIALS
www.afm-journal.de

Machine Learning-Evolutionary Algorithm Enabled Design for 4D-Printed Active Composite Structures

*Xiaohao Sun, Liang Yue, Luxia Yu, Han Shao, Xirui Peng, Kun Zhou, Frédéric Demoly, Ruike Zhao, and H. Jerry Qi**

Active composites consisting of materials that respond differently to environmental stimuli can transform their shapes. Integrating active composites and 4D printing allows the printed structure to have a pre-designed complex material or property distribution on numerous small voxels, offering enormous design flexibility. However, this tremendous design space also poses a challenge in efficiently finding appropriate designs to achieve a target shape change. Here, a novel machine learning (ML) and evolutionary algorithm (EA) based approach is presented to guide the design process. Inspired by the beam deformation characteristics, a recurrent neural network (RNN) based ML model whose training dataset is acquired by finite element simulations is developed for the forward shape-change prediction. EA empowered with ML is then used to solve the inverse problem of finding the optimal design. For multiple target shapes with different complexities, the ML-EA approach demonstrates high efficiency. Combining the ML-EA with computer vision algorithms, a new paradigm is presented that streamlines design and 4D printing process where active straight beams can be designed based on hand-drawn lines and be 4D printed that transform into the drawn profiles under the stimulus. The approach thus provides a highly efficient tool for the design of 4D-printed active composites.

1. Introduction

Active composites are novel functional materials that consist of active materials with different properties and thus generate active responses to external stimuli. Upon the stimulus, such as heat,[1] light,[2] water,[3] and magnetic field,[4] the responding property mismatch of constituent materials endow the active composites with unique shape-changing behaviors that depend on the spatial distributions of materials or properties. As shown in **Figure 1**a, in general, a bilayer structure with a certain property mismatch can be actuated to bend, which has been exploited to achieve various shape changes.[5] Integrating active composites with 3D printing technology, which is capable of high-resolution placement of different materials, allows the 3D printed parts to transform their shapes in the 4th dimension, time, thus enabling a rapidly emerging technology of 4D printing.[6] Utilizing complex property distribution in a voxelized structure provides greater design flexibility for attainable shape changes for 3D/4D printing technology, thereby offering enormous design potentials.[6h] However, rationally designing the property distribution to achieve desired complex shape change is a great challenge[6h] due to the tremendous design space on numerous voxels, which is unlikely to be explored by experiments or be leveraged by intuitive design strategies that rely heavily on personal experience or knowledge.

Non-intuitive design strategies based on modeling or simulation approach, which can accommodate more trial-and-error cycles than experiments, have been widely used for the voxel-based design of active composites.[3,7] Here, a complete design process requires solving both the forward problem of predicting shape changes for given material or property distributions and the inverse problem of finding the optimal material or property distribution to obtain the desired shape change. The former often relies on accurate numerical models (or predictive models), such as

X. Sun, L. Yue, L. Yu, X. Peng, 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

H. Shao
Visual Computing Center
King Abdullah University of Science and Technology
Thuwal 23955–6900, Saudi Arabia

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/adfm.202109805.

DOI: 10.1002/adfm.202109805

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

F. Demoly
ICB CNRS 6303
Univ. Bourgogne Franche-Comté
UTBM
Belfort 90010, France

R. Zhao
Department of Mechanical Engineering
Stanford University
Stanford, CA 94305, USA

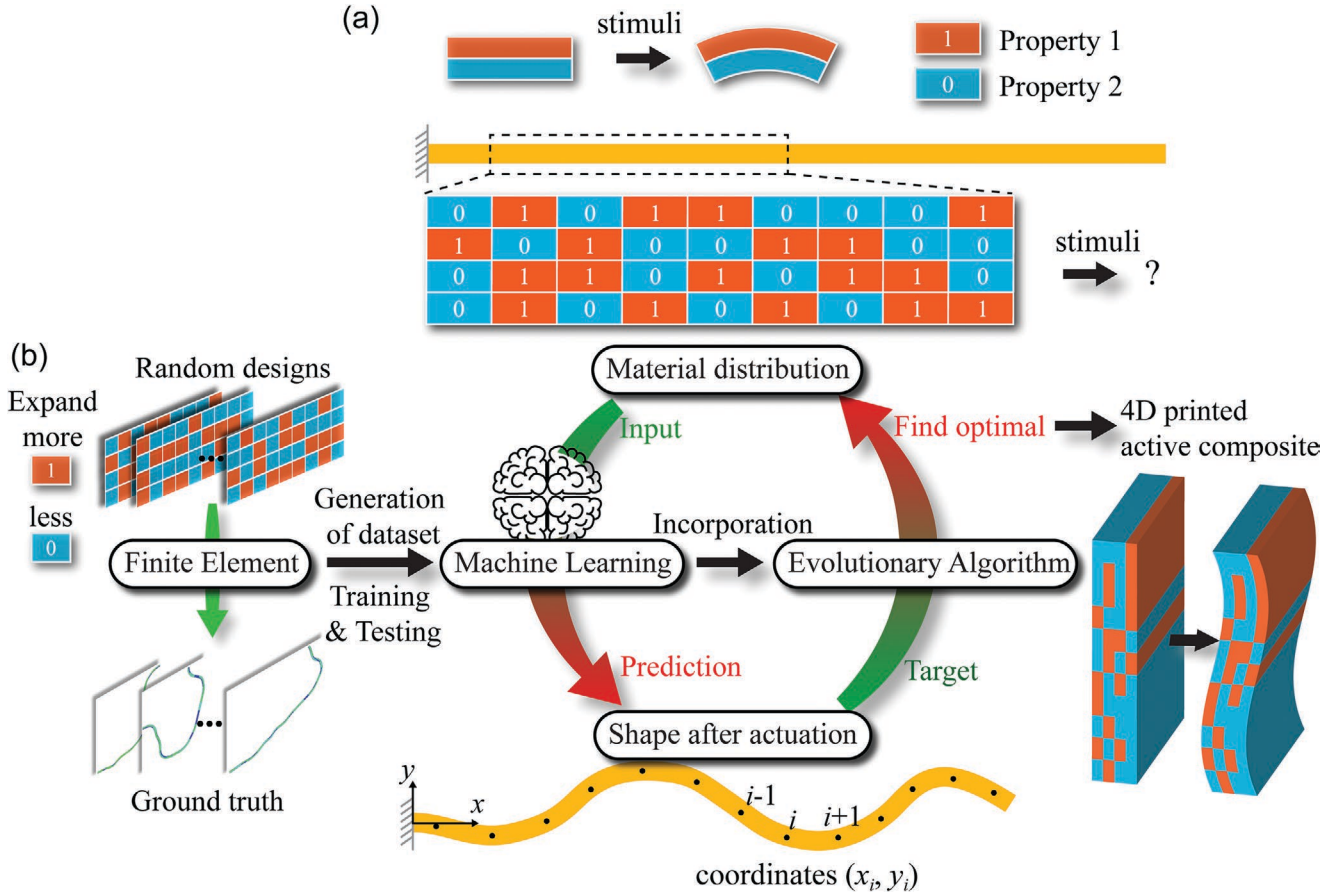


Figure 1. Schematic illustration of the proposed solution for the design of a 4D-printed active composite beam. a) Property mismatch-induced actuation of active composite with bilayer- or more complex property distribution. Two properties, "1" and "2", are encoded as "1" and "0", respectively. b) Complete design process: the dataset generation by FE simulations, the shape-change prediction by ML, and the material-distribution design by ML-integrated EA. The volumetric expansion mismatch is used to mimic a general eigenstrain mismatch induced by different mechanisms. Undeformed cantilever composite beam with a voxel-based material/property distribution that is digitally encoded into a 2D number array and is used as the input for the ML model. The actuated beam shape is parameterized as coordinate data of sampling points and is used as the output for the ML model.

finite element (FE) models, while the latter is often handled by incorporating the forward predictive model into some optimization algorithms. For example, topology optimization (TO)[8] based on different methods has been used to guide the design for 4D printing, such as optimizing compliances of soft actuators[9] or designing shape-changing behaviors of active composites.[10] Such gradient-based TO, however, may encounter difficulties when the active material involves higher geometric or material nonlinearities (e.g., multiphysics driven material nonlinearity). Alternatively, the method that integrates FE and other gradient-free optimization algorithms (such as evolutionary algorithms, EA), has achieved great success in designing certain shape-changing responses of active composites[11] or other engineering structural problems.[12] Such an evolutionary method generally relies on numerous iterations of FE simulations to explore a large design space, thus suffering from high computational cost. For example, Hamel et al.[11a] utilized the FE-based evolutionary algorithm (FE-EA) in solving the voxel-based inverse design problem of an active composite beam, but the optimization is time-intensive and cannot deal with too complicated target shapes (e.g., sinusoidal shape with ≥1 periods). Accurately and efficiently exploring a large design space and tackling inverse problems remain to be challenging.

Recent advancements in machine learning (ML)[13] offer new possibilities for developing fast, computationally affordable, and high-fidelity predictive models that can be integrated with the optimization algorithm to achieve an efficient inverse design. Existing works mainly focused on utilizing ML-guided design strategy for optimizing or predicting mechanical properties of materials, such as strength and toughness of composites,[14] stress and strain fields of composites,[15] Poisson's ratio of auxetic metamaterials,[16] responses of soft pneumatic robots,[17] among other material responses.[18] However, there is limited work on using ML-guided design for actuation or shape change response of active composites. Recently, Zhang et al.[19] utilized multiple ML models for predicting the shape change of voxelized active composites, and found the convolutional neural network (CNN) offered the best accuracy. However, their ML models were not used for the inverse design problem of material distributions for given target shape changes, probably because the CNN model cannot predict some complicated designs very well whilst the inverse design problem requires high prediction accuracy.

Motivated by the challenges in design for 4D printing,[6h] this work presents an efficient approach for both forward shape-change prediction and inverse material design for voxelized active composite beams, as illustrated in Figure 1b, by combining a recurrent neural network (RNN)-based ML and an EA. The volumetric expansion mismatch is used to mimic a general eigenstrain mismatch induced by different mechanisms. The RNN ML model is trained by the dataset acquired from FE simulations and shows high accuracy in predicting shape change with complex material distributions. The ML is then integrated with EA (ML-EA) for the inverse problem of finding the optimal design of property distribution based on a desired (or target) shape change. The fast and accurate ML model allows the EA to solve inverse problems in a very large design space that is impossible to be explored by FE simulations. As a result, optimal designs in terms of property distributions for multiple target shapes with different complexities are rapidly achieved. This highly efficient ML-EA design approach also permits a new streamlined design-fabrication paradigm where the computer vision (CV)-integrated ML-EA allows a quick fabrication of 4D-printed beams with shape changes based on hand-drawn lines.

2. Results

2.1. Machine Learning Model for Deformation Prediction

In this section, we present an RNN-based ML model that can accurately predict the nonlinear actuated deformation of an active composite beam. The composite beam consists of two materials, an active material with a large expansion and a passive material with a low expansion, which are encoded as "1" and "0", respectively, as shown in Figure 1b. The two materials are randomly assigned to $N_x \times N_y$ voxels, with N_x being the voxel number in the x- (length) direction and N_y in the y- (thickness) direction. The material distribution can therefore be digitally encoded into a 2D number array of "1's and" 0's (Figure 1b), which will be the input data for the ML model. Under a certain stimulus, the active material expands more while the passive material expands less, resulting in a shape change of the beam (Figure 1b). Such shape change can be parameterized as number arrays using coordinates (x_i, y_i) of certain points of the deformed beam, which will be the output data for the ML model. For the studied active beam, there exist 2^{NxNy} possible material distributions, implying a large and complex design space.

The FE model for the active composite beam, as described in Experimental Section, is developed to generate the dataset for the ML model. Random material distributions (or designs) are created, and the corresponding true shape changes are obtained by performing FE simulations (Figure 1b). The input and output data are then used to train the ML model to be constructed below.

An RNN-based ML model is used to predict the shape change of an active beam. This is inspired by the spatially sequential characteristics of the voxel-based material inputs and the sampled coordinate outputs in the length direction of the beam. RNN is a deep learning network well suited for dealing with sequential data, thus gaining wide popularity in tasks where the sequence of the data is essential, such as natural language processing. In recent years, RNN has been used in predicting various path-dependent mechanical responses of materials,[20] such as plastic flow,[20a] and fracture process.[20b] The capability of RNN for handling sequential data is because it uses the past information to predict the response of the current and future inputs. In an RNN, as shown in **Figure 2**a, an input I_t is acted on by the hidden state H_{t-1} to produce an output O_t. Certain information is stored in the hidden state (H_t) and passed to the next step. Repeating the process allows information flows from the past to the future. It is seen that the output, which relies on the current input and hidden state, depends on previous inputs as well. Different time steps (or blocks) of RNN share the same weights (parameters to be learned) that determine the hidden state and output of a specific step.

To use RNN, a similar data structure for the beam deformation problem is utilized. As shown in Figure 2b, a column (y-direction) of voxels is treated as a box whose state is represented by a single number-array input (size N_y). Therefore, the beam is comprised of N_x boxes and a material distribution can now be treated as a sequence of single-box inputs. In other words, for a given material distribution with voxel numbers of $N_x \times N_y$, the entire input is restructured into sequential inputs of length N_x with a single input (or box) having N_y features and thus 2^{Ny} possible states. Similarly, a sequence of single-box outputs can be obtained by sampling coordinates (x, y) from the middle point of each box, thus a single output has two features representing the coordinate values. Since the left end of the beam is fixed, the output of a box is dependent on all the "past" inputs (i.e., those from boxes on the left of the current box). More specifically, a current box output (x_i, y_i) relies directly on the current input s_i (i.e., local material distribution of the box), as well as the output (x_{i-1}, y_{i-1}) of the previous box, which essentially represents a hidden state (Figure 2b). Evidently, the beam shape data indicate a sequential characteristic in length (Figure 2b) identical to that of an RNN (Figure 2a), thus motivating us to use an RNN for the shape-change prediction. Our network architecture consists of a sequence input layer, a long short-term memory (LSTM)[21] network layer, a fully connected layer, and a regression layer. The LSTM is a special type of RNN that addresses the issues of vanishing or exploding gradients presented in long sequences.

The loss function is defined as the half-mean-squared-error between the predicted response (i.e., coordinate x_i or y_i) and true response (x_i^{true} or y_i^{true}) for an output sequence, that is,

$$\text{Loss}(x) = \frac{1}{2N_x}\sum_{i=1}^{N_x}\left(x_i - x_i^{\text{true}}\right)^2,\ \text{Loss}(y) = \frac{1}{2N_x}\sum_{i=1}^{N_x}\left(y_i - y_i^{\text{true}}\right)^2 \quad (1)$$

for coordinate x and y, respectively. More details on the construction and training of the RNN model are described in the Experiment Section.

Three cases with different voxel numbers ($N_x \times N_y$) are considered: 40×4, 24×4, and 40×2, which has the design space of 2^{160} ($\approx 1.46 \times 10^{48}$), 2^{96} ($\approx 7.92 \times 10^{28}$), and 2^{80} ($\approx 1.21 \times 10^{24}$), respectively. The size of the entire dataset is chosen to be 18 000 for the 40×4 case, 8600 for the 24×4 case, and 8000 for the 40×2 case. For each of the cases, the dataset is split

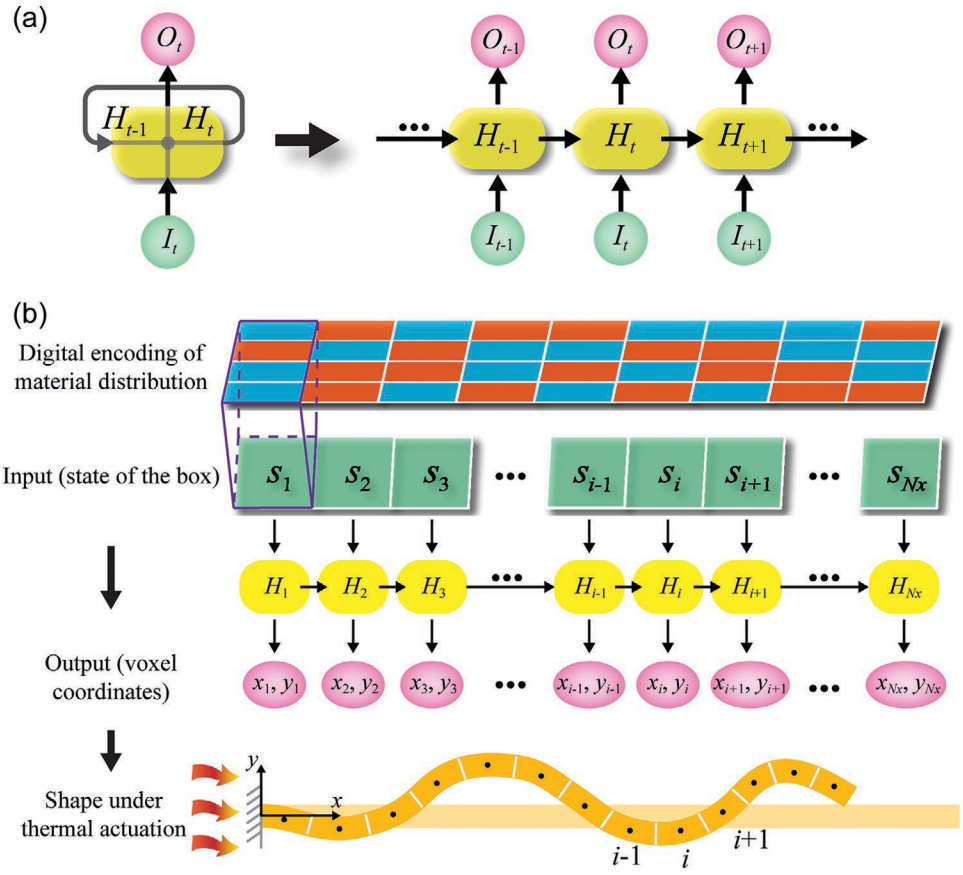


Figure 2. Motivation and design of the RNN-based ML model. a) Architecture of an RNN. b) Inputs and outputs of our problem with a similar structure to that of RNN.

into training, validation, and testing datasets with fractions of 0.7, 0.15, and 0.15. Statistics investigations show that the three datasets (training, validation, and testing) follow similar distributions (see Figure S1 and associated text in Section S1, Supporting Information). In addition, different amounts of data are randomly selected from the entire training dataset for model training to study the sensitivity of network performance to training set size. More details are provided in the Experiment Section.

2.2. Performance of Machine Learning Model

We first consider the case with $N_x = 24$, $N_y = 4$. **Figure 3**a shows the sensitivity of the network validation loss (defined in Equation (1) in Experimental Section) to the training set size. The size of 2000 is sufficiently large to give the converged results, and the converged validation loss is similar between coordinate x and y. With the entire training set (6000 data points), the training and validation losses versus epochs during the training process are shown in Figure 3b. The validation loss converges at epoch 55, before which no underfitting or overfitting is observed. Figure 3c shows comparisons of ground-truth shapes and ML-predicted shapes for 25 randomly picked data points from the test set. An excellent agreement is achieved between the ground-truth and predicted shapes. For the case with $N_x = 40$, $N_y = 4$, a larger training set size, 4000, is needed to achieve convergence in the validation loss (Figure S2a, Section S2, Supporting Information). The converged validation losses for both x and y are slightly lower than those for the 24×4 case. With the entire training set (12 800 data points), the training and validation losses versus epochs of training are given, showing no underfitting or overfitting before epoch 100 (Figure S2b, Section S2, Supporting Information). An excellent agreement can be seen between the ground-truth and ML-predicted shapes for 25 randomly picked data points from the test set (Figure S2c, Section S2, Supporting Information). In addition to the above two cases, the case with $N_x = 40$, $N_y = 2$ is also studied, in which the input data is further restructured to speed up the training process without losing accuracy (see Figure S3 and associated discussions in Section S2, Supporting Information). The results for all three cases demonstrate that RNN can achieve very high accuracy in predicting the shape change.

Further, we compare the performance of the RNN and a CNN for the 24×4 case. As shown in Figure S4a, Section S3, Supporting Information, our CNN architecture is similar to that of the regression CNN used in Zhang et al.[19] With the current dataset (total size 8600), RNN achieves better performance than CNN as shown in shape change comparisons for randomly picked data points and the regression of the ground-truth versus predicted values (Figure S4b,c, Section S3, Supporting Information). More specifically, CNN achieves the

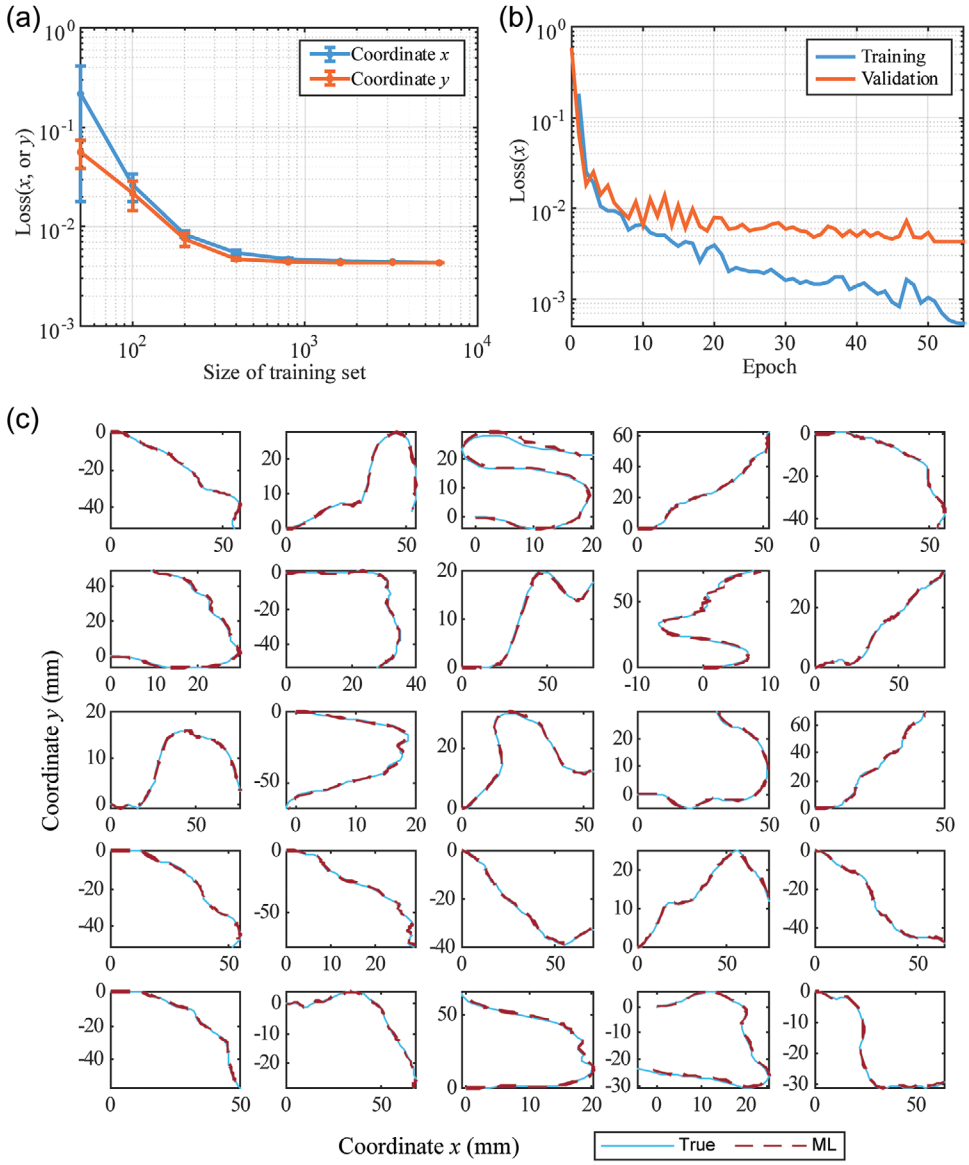


Figure 3. ML prediction for the case of 24×4 voxels. a) Sensitivity of the validation loss to the training set size. b) Training and validation loss as a function of epochs showing the training process. c) Comparison of true shapes and ML-predicted shapes for 25 random datapoints randomly picked from the test dataset.

$R^2 \approx 0.9$ implying a reasonably good prediction accuracy, while RNN achieves $R^2 > 0.999$ implying an excellent accuracy. This is expected since there is a strong similarity in the sequential data structure between the RNN and beam deformation problem, while CNN utilizes convolutional filters and pooling layers to recognize spatial features from the input which have no intuitive relation to the deformed beam coordinates. Also, since each coordinate output depends on the global voxel information in CNN, it may be easily disturbed by the voxel that is irrelevant.

In addition to the accuracy, The prediction speed of ML (RNN) against FE is also examined. For each $N_x \times N_y$ case, we perform benchmark tests on the time cost of ML and FE for 1000 shape predictions of randomly generated designs using one CPU core (Intel Core i9-10900) and one GPU (NVIDIA Quadro P620). As shown in **Table 1**, for all the cases, the ML prediction (on the order of seconds) is much faster than FE (on the order of hours). Results of FE time cost for 24×4 case will be used to estimate time cost for the FE-EA (Section 2.4 of the main text).

2.3. Evolutionary Algorithm Design based on Machine Learning Model

Next, we use the EA approach to design the material distribution for target actuated shapes. EA is a population-based stochastic search technique that utilizes the principles of natural selection to seek optimal inputs producing desired outputs.[22] As schematically illustrated in **Figure 4**a, the EA procedure

Table 1. Time cost for 1000 shape predictions with ML (RNN) and FE for different voxel numbers: "s" for seconds and "h" for hours. The estimation for time cost is based on one CPU core (Intel Core i9-10900) and one GPU (NVIDIA Quadro P620).

Time cost for 1000 predictions		
Voxel number $N_x \times N_y$	ML (RNN)	FE
24×4	6.5s	14h
40×2	9.11s	14h
40×4	17.4s	15.5h

starts by creating a population of random candidate solutions (i.e., material distribution design as digitally encoded into the array of "1's and "0's). The candidate solutions are called individuals of the population and the digital encoding can be seen as the individual's genes. The EA then iteratively evaluates and evolves the population over successive generations, in which good individuals survive and reproduce whilst bad individuals are eliminated, until the number of generations reaches N_{gen} or an acceptable solution is found. In our approach, in each generation, individuals are evaluated by using the well-trained ML model to predict the actuated shapes and calculating fitness values for the individuals. The general form of fitness function can be defined as

$$f_1 = \sqrt{\frac{1}{N_x}\sum_{i=1}^{N_x}\left[\left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2\right]} \tag{2}$$

which is the root-mean-squared (RMS) error between the ML-predicted coordinates (x_i, y_i) and target coordinates $(\hat{x}_i, \hat{y}_i)$ for sampling points (mid-point of each column of voxels). Here, the fitness value is a measure of how close the actuated shape is

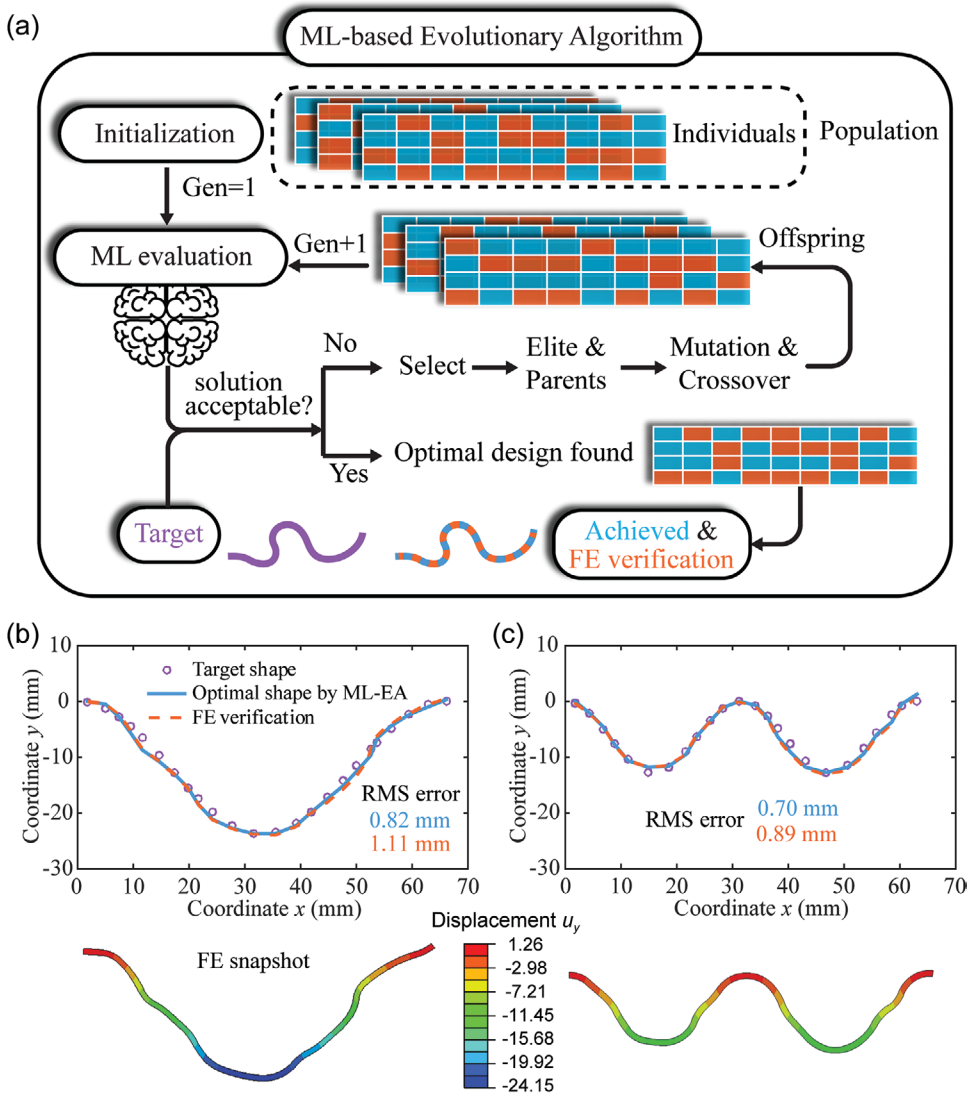


Figure 4. Demonstration of ML-based EA approach. a) Schematic of ML-EA optimization approach. b) One-period target shape, ML-EA optimal shape, and corresponding FE shape. c) Two-period target shape, ML-EA optimal shape, and corresponding FE shape. The color map illustrates the y-displacement in mm.

to the target shape, and the individual with a lower fitness value is more favored. Therefore, the goal of the optimization is to minimize the fitness function essentially.

After evaluation, the EA creates an offspring population for the next generation based on the current population. More details are provided in Experimental Section. The evolutionary process is then repeated, and the optimal solution in each generation is recorded. Next, we will evaluate the capability of the ML-EA approach using multiple target shapes. The case with a voxel number of 24 (N_x) × 4 (N_y) will be focused on.

As the first example, the one-period and two-period sinusoidal target shapes are considered. Due to the large deformation, it is not easy to properly specify coordinates $(\hat{x}_i, \hat{y}_i)$ of sampling points for the target shape. To circumvent this issue, the following fitness function, which uses the ML-predicted coordinates (x_i, y_i) only, is adopted to evaluate the candidate solutions,

$$f_2 = \sqrt{\frac{1}{N_x}\sum_{i=1}^{N_x}\left[\frac{y_i}{|y_i|_{\max}} - \frac{1}{2}\cos\left(2n\pi\frac{x_i}{|x_i|_{\max}}\right) + \frac{1}{2}\right]^2} \tag{3}$$

The number of periods, n, is taken as 1 and 2 for the two target shapes, respectively. Note that f_2 vanishes when the beam shape satisfies the sinusoidal profile. With the fitness function Equation (3), the target coordinates $(\hat{x}_i, \hat{y}_i)$ are not strictly prescribed but can be reconstructed as described in Experimental Section.

The following EA parameters are fixed for the optimization: population size (number of individuals) = 500 and N_{gen} = 300. Figure 4b,c shows the optimization results for cases with one period (n = 1) and two periods (n = 2), respectively, where good agreement is achieved between the target shape (symbols) and the ML-EA optimal shape (solid lines). FE simulations for the optimal material designs are finally performed, and the FE-predicted shapes (dashed lines) verify the accuracy of the ML-predict ones (solid lines). To further quantify the accuracy of the optimal design, we calculate the RMS errors using Equation (2) of both ML- (blue) and FE- (orange) predicted shapes of the optimal design against the target shape. Note that the EA iterations directly use the ML predictions whose error will be inevitably forwarded to the EA process. Nonetheless, excellent agreement is achieved between the optimal and the target shapes, demonstrating the high performance of the ML-EA approach. In addition, the ML-EA achieves the optimal design in ≈11 min by using ≈150th generations, which require roughly 75 000 evaluations of different material designs. It can be inferred that for the same target shapes, the FE-EA[11a] would consume more than 1000 h for both two cases, as summarized in **Table 2**, which are based on the benchmark time cost for 1000 FE predictions (see Table 1).

2.4. Distance-Weighted Fitness Function for More Complicated Shapes

We now consider two more complicated target shapes. The first target is a three-period shape composed of twelve quarter-circles. Using the same ML-EA approach with the fitness function

Table 2. Computational cost for the ML-EA and FE-EA models: "m" for minutes and "h" for hours. The estimation for time cost is based on one CPU core (Intel Core i9-10900) and one GPU (NVIDIA Quadro P620). The RMS errors are calculated using Equation (2).

	EA time cost (24 × 4 voxels)		
Design	ML-EA	FE-EA	RMS errors
One-period:	11 m	1053 h	0.82 and 1.11 mm
Two-period:	11 m	1053 h	0.70 and 0.89 mm
Three-period:	54 m	5263 h	1.37 and 1.34 mm
Half-butterfly:	22 m	2105 h	0.96 and 1.02 mm

given by Equation (2), optimal shapes with different population sizes are obtained (see Figure S5, Section S4, Supporting Information). A large discrepancy between the target and the achieved shape is observed even after evolving for 200 generations with 5000 populations per generation (i.e., one million shape evaluations performed). The reason for the large discrepancy is illustrated below using the population size of 1000 as an example (Figure S5, Section S4, Supporting Information). In general, a large difference is observed on the left half part of the beam. During the evolutionary process, a potential improvement (i.e., shape change) in the left half beam can lead to a large shape change and thus discrepancy on the right half. This may increase the fitness function, resulting in that the solution being discarded and there is no chance to evolve to the globally best solution. This is due to the boundary condition of our problem where the left end is fixed and the right end is free. Essentially, the found solution represents a local minimum in the search space and it would be very hard for the solution to jump out to the global minimum using the current fitness function. To resolve this problem, a distance-weighted fitness function is adopted, which is

$$f_3 = \sqrt{\frac{1}{N_x}\sum_{i=1}^{N_x}\frac{k_x\left(x_i - \hat{x}_i\right)^2 + k_y\left(y_i - \hat{y}_i\right)^2}{i}} \tag{4}$$

Here, the box number i represents a topological distance of the box from the starting point (i.e., left end), k_x and k_y are introduced to regulate the weights of errors in two coordinates. With the modified fitness, the EA will favor the optimization of the beam shape sequentially from left (smaller distance) to right (larger distance), then an improvement in the distant part will not affect the shape change in the nearer part, thereby helping the population evolve towards the global minimum.

With this new fitness function of Equation (4), optimal designs for the three-period shape (**Figure 5**a,b) and the half-butterfly shapes (Figure 5c) are successfully achieved. For the three-period target shape, Figure 5a shows the sensitivity of the optimization process and optimal solution to the population size. The RMS errors (based on Equation (2)) of the true (or FE) shape of the optimal design against the target shape are provided. Starting from the population size of 2500, the optimal solution is acceptable, which is achieved in around 150th generations. In this case, roughly 375 000 evaluations are performed for the optimization, which is much higher than the size of

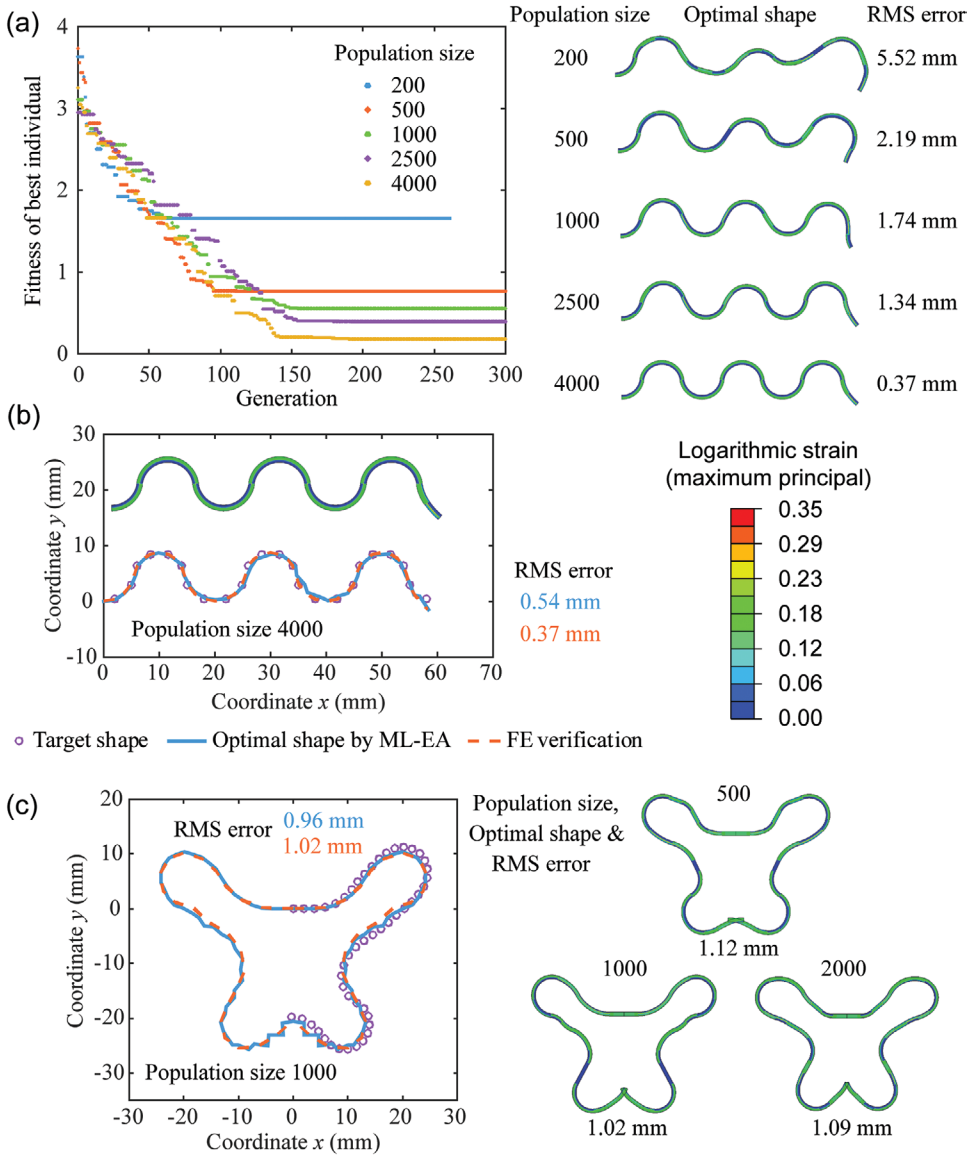


Figure 5. ML-EA optimization for a,b) more complicated three-period shape and c) half-butterfly shape. (a) Evolutionary process and optimal shapes for different population sizes. (b) Target-shape, ML-EA optimal shape, and corresponding FE shape (true shape). (c) Target-shape, ML-EA optimal shape, and corresponding FE shape (true shape). The optimal shapes for different population sizes are shown on the right.

the training dataset (≈1000–10 000). Figure 5b shows excellent agreement among the target shape (symbol), the ML-EA optimal shape (solid line), and the FE-achieved true shape of the optimal design (dashed line). The RMS errors of both the true shape (orange) and ML-predicted shape (blue) of the optimal design against the target shapes are provided. As we mentioned earlier, although the error of the ML predictions will be forwarded to the EA process, excellent performance of the ML-EA is demonstrated: the error of ground-truth shape of the optimal design against the target can be even lower than that of the ML-predicted one (Figure 5b). Figure 5c shows the results for the final demonstration, the half-butterfly shape, where the optimal shape (solid line) again achieves great agreement with the target (symbols) and the true shape by FE (dashed line). Optimal shapes with different population sizes are shown on the right. The RMS errors between the optimal and target shapes are provided as well.

Using the number of shape evaluations of an entire EA process, the computational cost for the cases with different target shapes is estimated and listed in Table 2. The time cost estimation for the FE-EA[11a] is based on the same number of shape evaluations that are needed in ML-EA and on one CPU core. The RMS errors of the ML-predicted (first) and ground-truth (second) shapes of the optimal design against the target shape for all the studied cases are provided for better quantification of the accuracy. As a quick summary, our ML-EA approach demonstrates very high efficiency in solving inverse material design problems for a complicated target shape. This is attributed to the use of the ML model in the shape evaluation of EA, which allows for a much more rapid search in a complex design space

compared to the FE-EA in the previous work.[11] In addition, the design for other different target shapes can be easily performed with no need for further FE simulations, which significantly reduces the computational time.

2.5. Rapid 4D Printing Design and Fabrication: 4D Printed Beam with Shape Changes based on Hand-Drawn Lines

With the high efficiency of our proposed ML-EA approach, we further integrate the ML-EA approach with CV algorithms to enable a new paradigm that streamlines the design and fabrication process for 4D printing, as schematically illustrated in **Figure 6**a. We first identify the target shapes from hand-drawn lines, that is, coordinates $(\hat{x}_i, \hat{y}_i)$, from the raw images with drawn profiles, using CV algorithms (see Experimental Section, and Figure S6 and associated text in Section S5, Supporting Information). Then the ML-EA approach is utilized to obtain the optimal designs, that is, spatial distributions of material phase "1" and "0" with mismatched expansion properties, that yield the target shapes. The 24×4 voxels, 1000 population size, and 150 evolutionary generations are used for the ML-EA design, and each shape design takes 22 min. To incorporate an optimal design into a real 3D-printed structure, we employ a photo-curable polymer that can expand through swelling and use the grayscale digital light processing (g-DLP)[23] technique to introduce two differentially polymerized material phases that exhibit expansion mismatch (see Experimental Section, and Figure S7 and associated text in Section S6, Supporting Information). In g-DLP, the optimal design is transformed into images with grayscale patterns for the DLP printing, which thus spatially assigns two distinct expansion properties as desired. Finally, the printed active beams are immersed in acetone to swell for shape transformation.

Figure 6b presents the results for five different shapes to demonstrate our new paradigm. Hand-drawn lines, identified target shapes with ML-EA designs, as-printed strips, and final actuated shapes are shown in four columns (left to right). In the second column, the obtained optimal designs are shown in grayscale patterns. The corresponding shapes predicted by ML (solid lines) and FE (dashed lines) agree well with the target shape (symbols), implying the great accuracy of the ML-EA approach. Experiments are further conducted to validate the ML-EA optimal designs. Note that the practical material properties (modulus and expansion) in experiments are different from those used in ML-EA (i.e., identical modulus and strain mismatch of 0.1 between the two phases). The issue can be resolved by re-training the ML model based on the FE data with practical material properties and re-running ML-EA. Here, we present an alternative strategy with no need for re-training to compensate effects of the property difference on the shape change. An analytical model is developed to convert the ML-EA designs based on practical material properties (see Experimental Section, and Figure S8 and associated text in Section S7, Supporting Information). The converted ML-EA designs are used for 3D printing. As shown in the two rightmost columns (Figure 6b), the printed strips transform their initially straight shapes into target shapes upon swelling. As a proof-of-concept design approach, the CV-integrated ML-EA demonstrates high efficiency in generating optimal designs to convert line drawings into true shape changes of 4D-printed beams.

2.6. Discussions

The ML-EA approach demonstrates high efficiency in solving inverse problems in a complex design space, that is, inversely designing the material distribution of active composite beams to obtain desired mechanical responses, which cannot be achieved by conventional methods such as FE-EA but is enabled by the rapid, accurate and computationally inexpensive ML model. Although thousands of FE simulations are needed to train the ML model, this number (1000–10 000) is much lower than that of the mechanical response evaluations in an EA process required to obtain an optimal design for complicated target shapes. Moreover, once the ML is trained, ML-EA can deal with different target shapes with no need for further FE simulations. In addition, although thermal expansion is utilized as the actuation mechanism in the FE model, the ML-EA is generally applicable to various material systems with different active strain mechanisms, such as swelling, shape memory, and magnetism, as long as the mismatched eigenstrain can be induced. As the material system changes, one may either re-train the RNN based on the FE data with practical material parameters or modify the optimal designs to approximately compensate effects of property difference between existing ML-EA designs and experiments using the approach described in Section S7, Supporting Information. The ML-EA may be also extended to the cases with multiple (>2) material phases in the beam which provides greater design flexibility. Therefore, our approach will be useful for motivating the design for various 4D-printed active composite beams.

It is worth noting that in this work, our ML-EA approach is focused on the design of 1D active composite beams. In this case, RNN is leveraged to deal with sequential data (of 1D dependency) arising from the beam problem, providing a highly efficient ML model. However, there is a great need for efficient design of 2D and 3D active structures based on desired complex shape change, which could be challenging due to the further increased design space and higher-dimensional data dependencies. For such a problem, the ML model capable of handling data with more complex spatial dependencies can be helpful and will be explored in our future work.

3. Conclusions

We present a novel approach for mechanical response prediction and inverse design of a 4D-printed active composite beam based on ML and EA. A RNN based ML model is utilized to predict the shape change based on the material distribution of the active beam. The ML model achieves excellent accuracy on the test set for three cases with different voxel numbers, 24×4, 40×4, and 40×2. In addition, the ML prediction is much more rapid and computationally inexpensive than the FE model. Regarding different ML models, the RNN achieves better performance than CNN in shape change predictions in the 1D cantilever setting. We then incorporate the trained ML

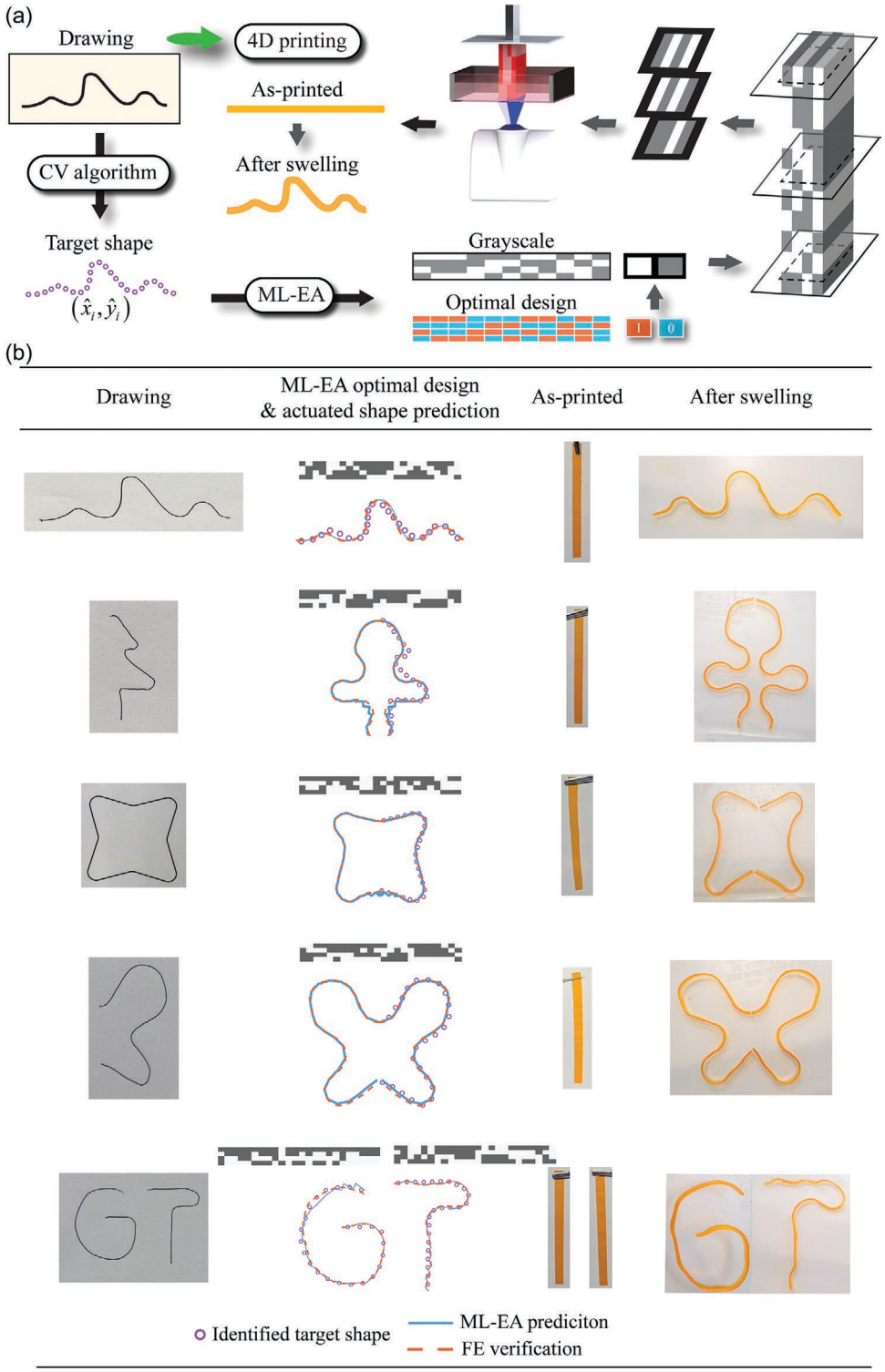


Figure 6. Integrating CV with ML-EA to streamline the design and fabrication. a) 4D printing design, which includes identifying drawn profiles as target shapes, performing the ML-EA design, converting the obtained optimal design to the grayscale slices, performing DLP 3D printing, actuating the printed structure by expansion. b) 4D printing to achieve different line drawings. From top to bottom: mountain, half-tree, four-pointed star, half-butterfly, "GT". From left to right: line drawing by hand, ML-EA optimal designs and predicted optimal shapes, as-printed shapes, and shapes after swelling.

(RNN) model into an EA for the inverse design of the material distribution based on the desired shape change. The ML-EA optimization is performed based on multiple target shapes with different complexities. The proposed distance-weighted fitness function greatly improves the capability to handle complex shape change. As a result, for all the target shapes, the ML-EA

rapidly achieves an optimal design of the material distribution, which is computationally verified by the excellent agreement among the desired shape, the ML-predicted shape of the optimal design, and the true shape of the optimal design achieved by FE. Finally, by integrating ML-EA with CV, we present a new paradigm that streamlines the design and fabrication process for 4D printing based on hand-drawn lines. The proposed CV algorithm enables the automatic identification of target shapes from raw images with line drawings. For multiple hand-drawn lines, the CV-ML-EA rapidly generates optimal designs and the 4D-printed active beams deform into drawn profiles under the stimulus.

4. Experimental Section

Finite Element Model for Data Generation: The FE simulations were conducted, using the commercial software ABAQUS (version 2018, Simulia, Providence, RI), to generate the dataset to be fed into the ML model. In the FE model, a cantilever composite beam under the 2D plane strain assumption and with the left end fixed was considered. Both active and passive materials were modeled using the incompressible neo-Hookean model with the same mechanical parameters. Expansion was achieved through the thermal expansion in the simulation, although it was not limited to thermal expansion. To achieve the expansion mismatch, the coefficient of thermal expansion was set to be 0.001 for active material and 0 for passive material. An environmental stimulus of 100 °C temperature increase was applied to the entire beam, upon which the active material phase was subjected to a linear strain of 0.1. The beam has a dimension of 80 mm long × 1 mm thick, which was meshed into $960 \times 12 = 11\,520$ hybrid plane strain (CPE4H) elements. Regarding the voxelization, the beam was partitioned into $N_x \times N_y$ voxels followed by the assignment of materials. Each voxel has $(960/N_x) \times (12/N_y)$ elements and the adjacent voxels shared the same boundary nodes. The entire FE model was automatically generated and run through a Python script. With the FE model, the dataset was generated by first creating random material distribution designs and then performing FE simulations to obtain the ground-truth shape change for each design configuration. The material distributions and the corresponding shape changes were inputs and outputs, respectively. The structures of the input and output are detailed in Section 2.1. The generated dataset was split into training, validation, and testing datasets with fractions of 0.7, 0.15, and 0.15. Statistics investigations on the inputs and outputs showed that three datasets (training, validation, and testing) follow similar distributions (see Figure S1 and associated text in Section S1, Supporting Information).

Construction and Training of the RNN Model: The network architecture consists of a sequence input layer, an LSTM[21] layer, a fully connected layer, and a regression layer. The implementation, training, and testing were conducted using Matlab (2020a, MathWorks, Natick, MA). Before the training, all the input and output data were normalized using the z-score method, that is, $x' = (x - \text{mean}(x))/\text{std}(x)$, where x and x' are the raw and normalized feature values, respectively, and mean is the mean value and std is the standard deviation. The randomly generated raw inputs (numerous "1" and "0") showed a mean value of 0.5 and a standard deviation of 0.5. As a result, the input state "0" and "1" become "−1" and "1" after normalization. Such normalization was found to improve the network performance, which could be explained by the contribution of the symmetric (y-direction) characteristic of the normalized inputs. Two networks were trained to predict coordinate x and y separately to better identify their respective errors. The LSTM utilizes a special structure consisting of multiple gates and four neural network layers to control the update of the hidden state. The hidden size (number of neurons of each neural layer) of LSTM was set as 50. In the case, a single input has the size N_y and output has the size 1. Therefore the LSTM layer has $200 \times (50 + N_y + 1)$ learnable parameters. Regarding the hyperparameters, the initial learning rate was set to 0.005, which decreases by multiplying a factor of $1/\sqrt{2}$ every 50 epochs. The training stops after the validation loss converges. The mini-batch size during training was set to 64 or 10% of the training set size, whichever was smaller. The adaptive moment estimation (Adam)[24] optimizer was used to train the network.

To study the sensitivity of network performance to training set size, different amounts of data were selected from the entire training dataset (i.e., 70% of the entire dataset) for model training. The size of the entire training set was denoted by $n_{\text{training\_entire}}$, while the size of the selected training set was denoted by n_{training} and was varied. For each $n_{\text{training}} \leq n_{\text{training\_entire}}$, different batches of training data were randomly picked from the entire training set and used to train multiple ML models (the smaller of 5 and $\text{ceil}(n_{\text{training\_entire}}/n_{\text{training}})$), whose averaged validation loss was used to evaluate the performance for the particular n_{training}. The model trained using the entire training set was adopted for the optimization problem.

Methods for the ML-EA Design: The procedure of evolving a population is schematically illustrated in Figure 4a. After evaluation of an entire population, the EA created an offspring population for the next generation based on individuals of the current population. Three types of "children" were created: elite (5%), crossover (76%), and mutation (19%) children. The top 5% fittest individuals of the current population were elite, which directly survive to the next generation. To create the other two types of children, the "parents", a group of better performing individuals determined via the binary tournament technique were selected, which contribute their genes to the offspring: crossover children were created by combining genes of pairs of parents, while mutation children were created by randomly changing the genes of single parents. The choices of crossover and mutation operators were following the approach by Deep et al.,[25] to enforce the genes to be integer-valued (0 or 1). The EA optimizations were performed using Matlab.

For the cases with sinusoidal target shapes as shown in Figure 4b,c, the fitness function Equation (3) was adopted. In this case, the target coordinates $(\hat{x}_i, \hat{y}_i)$ were not strictly prescribed but can be reconstructed through

$$\hat{x}_i = x_i, \hat{y}_i = |y_i|_{\max}\left[\frac{1}{2}\cos\left(2n\pi\frac{x_i}{|x_i|_{\max}}\right) - \frac{1}{2}\right] \tag{5}$$

The reconstructed target shape depended on the $|x_i|_{\max}$ and $|y_i|_{\max}$ of a specific individual and was therefore not unique. However, since the reconstructed shape satisfied the sinusoidal profile of certain periods, it was used as a benchmark against which the RMS errors based on Equation (2) were computed to quantify the quality of actuated shapes of certain optimal designs.

Computer Vision Algorithms for Target Shape Identification: The raw image with hand-drawn lines was first processed using the Canny edge detection algorithm[26] in Matlab and converted to a black-and-white binary image where the white pixels form the edge of the drawing (Figure S6a, Section S5, Supporting Information). These white pixels contain no sequential information to define a path, so they cannot be directly used as a target shape, that is, coordinates $(\hat{x}_i, \hat{y}_i)$, for the ML-EA. Therefore, an arc-intersection tracing algorithm was developed that can extract the target shape from the binary image. The algorithm is described in Figure S6b and associated text in Section S5, Supporting Information. The algorithm enables the automatic identification of target shapes from raw images. More details can be found in Section S5, Supporting Information.

Materials and 4D Printing: The photo-curable resin for printing was a mixture of 2-Hydroxyethyl Acrylate (Sigma Aldrich, USA) and N-Isopropylacrylamide (Sigma Aldrich) with the weight ratio of 1:1. Photoinitiator (Irgacure 819, Sigma Aldrich) and photo absorber (Sudan I, Sigma Aldrich) were added additionally. The resin was well mixed and degassed before printing. A homemade bottom-up DLP-based 3D printer[23] was used to print the designed structure. The

Table 3. Parameters for 3D printing with CLIP and g-DLP techniques.

Grayscale percentage and light intensity	Irradiation time per layer	Thickness per layer	Pixel size
0%: 17.1 mW-cm^{-2}	3s	50 μm	50 μm × 50 μm
60%: 4.2 mW-cm^{-2}			

continuous liquid interface production (CLIP)[27] technique was utilized to enable fast printing speed. The grayscale-DLP technique was utilized to introduce two differentially polymerized material phases into the structure (Figure S7, Section S6, Supporting Information). The 3D printing parameters are listed in **Table 3**. The two phases exhibit a volumetric expansion mismatch upon swelling in acetone or water, similar to that considered in ML-EA (see discussions in Section S6, Supporting Information). The ML-EA designs were transformed into printing slices with grayscale patterns, where the active phase ("1") and passive phase ("0") were assigned with grayscale percentages 0% and 60%, respectively (Figure 6a). Upon layer-by-layer ultraviolet projecting of obtained slices, the grayscale pattern can spatially tune the light intensity and degree of conversion (DoC), thereby assigning two distinct volumetric straining properties in the printed polymer as designed. More details can be found in Section S6, Supporting Information.

Note that the practical material properties in experiments were different from those used in ML-EA. As shown in Section S7, Supporting Information, the printed two material phases with different DoC show a modulus ratio of 0.15, while the ML-EA design assumed the identical modulus for two constituent phases. The practical expansion mismatch was identified to be 0.072, which was also different from that used in ML-EA (i.e., 0.1). Such issues can be resolved by re-training the ML model based on the FE data with practical material properties (expansion mismatch and modulus difference) and re-running the ML-EA. Here, with no need for re-training, an alternative strategy was adopted to approximately compensate effects of difference in modulus and expansion mismatch of the two phases on the shape change, that is, converting the optimal designs based on analytical curvatures of multi-layer composite beams (see Figure S8 and associated text in Section S7, Supporting Information). FE simulations were further performed, and the obtained shape changes based on original and converted optimal designs were consistent with the experimental observations (Figure S8, Supporting Information), validating the design conversion strategy. Experimental shapes in Figure 6b were based on the converted optimal designs, which were also consistent with the FE results (Figure S9, Supporting Information). More details can be found in Section S7, Supporting Information.

Supporting Information

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

Acknowledgements

H.J.Q. acknowledges the support of an AFOSR grant (FA9550-20-1-0306; Dr. B.-L. "Les" Lee, Program Manager) and a gift fund from HP, Inc. X.S. thanks Xingwei Yang for helpful discussions.

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, evolutionary algorithms, machine learning

Received: September 28, 2021
Revised: November 4, 2021
Published online: November 21, 2021

[1] a) J. T. Wu, C. Yuan, Z. Ding, M. Isakov, Y. Q. Mao, T. J. Wang, M. L. Dunn, H. J. Qi, *Sci. Rep.* **2016**, *6*, 24224; b) Y. Q. Mao, K. Yu, M. S. Isakov, J. T. Wu, M. L. Dunn, H. J. Qi, *Sci. Rep.* **2015**, *5*, 13616; c) D. J. Roach, X. Kuang, C. Yuan, K. Chen, H. J. Qi, *Smart Mater. Struct.* **2018**, *27*, 125011.

[2] a) A. Lendlein, H. Jiang, O. Jünger, R. Langer, *Nature* **2005**, *434*, 879; b) H. Yang, W. R. Leow, T. Wang, J. Wang, J. Yu, K. He, D. Qi, C. Wan, X. Chen, *Adv. Mater.* **2017**, *29*, 1701627.

[3] a) A. S. Gladman, E. A. Matsumoto, R. G. Nuzzo, L. Mahadevan, J. A. Lewis, *Nat. Mater.* **2016**, *15*, 413; b) Z. A. Zhao, X. Kuang, C. Yuan, H. J. Qi, D. N. Fang, *ACS Appl. Mater. Interfaces* **2018**, *10*, 19932.

[4] S. M. Montgomery, S. Wu, X. Kuang, C. D. Armstrong, C. Zemelka, Q. J. Ze, R. D. Zhang, R. K. Zhao, H. J. Qi, *Adv. Funct. Mater.* **2020**, *31*, 2005319.

[5] Z. Zhao, J. T. Wu, X. M. Mu, H. S. Chen, H. J. Qi, D. N. Fang, *Macromol. Rapid Commun.* **2017**, *38*, 1600808.

[6] a) Q. Ge, H. J. Qi, M. L. Dunn, *Appl. Phys. Lett.* **2013**, *103*, 131901; b) S. Tibbits, *TED Talks* **2013**; c) X. Kuang, D. J. Roach, J. T. Wu, C. M. Hamel, Z. Ding, T. J. Wang, M. L. Dunn, H. J. Qi, *Adv. Funct. Mater.* **2019**, *29*, 1805290; d) S. M. Montgomery, X. Kuang, C. D. Armstrong, H. J. Qi, *Curr. Opin. Solid State Mater. Sci.* **2020**, *24*, 100869; e) B. Peng, Y. Yang, K. A. Cavicchi, *Multifunct. Mater.* **2020**, *3*, 042002; f) M. Bodaghi, A. R. Damanpack, W. H. Liao, *Smart Mater. Struct.* **2016**, *25*, 105034; g) M. Bodaghi, W. H. Liao, *Smart Mater. Struct.* **2019**, *28*, 045019; h) F. Demoly, M. L. Dunn, K. L. Wood, H. J. Qi, J.-C. André, *Mater. Des.* **2021**, *212*, 110193.

[7] a) Z. Zhao, H. J. Qi, D. N. Fang, *Soft Matter* **2019**, *15*, 1005; b) G. Sossou, F. Demoly, H. Belkebir, H. J. Qi, S. Gomes, G. Montavon, *Mater. Des.* **2019**, *175*, 107798; c) G. Sossou, F. Demoly, H. Belkebir, H. J. Qi, S. Gomes, G. Montavon, *Mater. Des.* **2019**, *181*, 108074.

[8] O. Sigmund, K. Maute, *Struct. Multidiscipl. Optim.* **2013**, *48*, 1031.

[9] A. Zolfagharian, M. Denk, M. Bodaghi, A. Z. Kouzani, A. Kaynak, *Acta Mech. Solida Sin.* **2020**, *33*, 418.

[10] K. Maute, A. Tkachuk, J. Wu, H. J. Qi, Z. Ding, M. L. Dunn, *J. Mech. Des.* **2015**, *137*, 111402.

[11] a) C. M. Hamel, D. J. Roach, K. N. Long, F. Demoly, M. L. Dunn, H. J. Qi, *Smart Mater. Struct.* **2019**, *28*, 065005; b) S. Wu, C. M. Hamel, Q. Ze, F. Yang, H. J. Qi, R. Zhao, *Adv. Intell. Syst.* **2020**, *2*, 2000060.

[12] a) G. X. Gu, L. Dimas, Z. Qin, M. J. Buehler, *J. Appl. Mech.* **2016**, *83*, 071006; b) A. Muc, W. Gurba, *Compos. Struct.* **2001**, *54*, 275; c) M. Abdi, R. Wildman, I. Ashcroft, *Eng. Optim.* **2014**, *46*, 628; d) K. Salonitis, D. Chantzis, V. Kappatos, *Int. J. Adv. Manuf. Technol.* **2017**, *90*, 2689.

[13] a) K. Guo, Z. Yang, C.-H. Yu, M. J. Buehler, *Mater. Horiz.* **2021**, *8*, 1153; b) Z. Jin, Z. Zhang, K. Demir, G. X. Gu, *Matter* **2020**, *3*, 1541.

[14] a) G. X. Gu, C.-T. Chen, D. J. Richmond, M. J. Buehler, *Mater. Horiz.* **2018**, *5*, 939; b) C.-T. Chen, G. X. Gu, *Adv. Theory Simul.* **2019**, *2*, 1900056.
[15] Z. Yang, C.-H. Yu, M. J. Buehler, *Sci. Adv.* **2021**, *7*, eabd7416.
[16] J. K. Wilt, C. Yang, G. X. Gu, *Adv. Eng. Mater.* **2020**, *22*, 1901266.
[17] A. Zolfagharian, L. Durran, S. Gharaie, B. Rolfe, A. Kaynak, M. Bodaghi, *Sens. Actuators, A* **2021**, *328*, 112774.
[18] a) D. J. Roach, A. Rohskopf, C. M. Hamel, W. D. Reinholtz, R. Bernstein, H. J. Qi, A. W. Cook, *Addit. Manuf.* **2021**, *41*, 101950; b) S. Rawat, M. Shen, *arXiv* **2018**.
[19] Z. Zhang, G. X. Gu, *Adv. Theory Simul.* **2020**, *3*, 2000031.
[20] a) M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, M. A. Bessa, *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 26414; b) Y.-C. Hsu, C.-H. Yu, M. J. Buehler, *Matter* **2020**, *3*, 197; c) L. Wu, V. D. Nguyen, N. G. Kilingar, L. Noels, *Comput. Methods Appl. Mech. Eng.* **2020**, *369*, 113234; d) H. J. Logarzo, G. Capuano, J. J. Rimoli, *Comput. Methods Appl. Mech. Eng.* **2021**, *373*, 113482.
[21] S. Hochreiter, J. Schmidhuber, *Neural Comput.* **1997**, *9*, 1735.
[22] a) D. E. Goldberg, *Genetic algorithms*, Pearson Education India, Delhi, India **2006**; b) K. Deb, in *Multi-objective evolutionary optimisation for product design and manufacturing*, Springer, Berlin, Germany **2011**, p. 3.
[23] a) X. Kuang, J. T. Wu, K. J. Chen, Z. Zhao, Z. Ding, F. J. Y. Hu, D. N. Fang, H. J. Qi, *Sci. Adv.* **2019**, *5*, eaav5790; b) Q. Zhang, X. Kuang, S. Weng, L. Yue, D. J. Roach, D. Fang, H. J. Qi, *Adv. Funct. Mater.* **2021**, *31*, 2010872.
[24] D. P. Kingma, J. Ba, *arXiv* **2014**.
[25] K. Deep, K. P. Singh, M. L. Kansal, C. Mohan, *Appl. Math. Comput.* **2009**, *212*, 505.
[26] J. Canny, *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *PAMI-8*, 679.
[27] J. R. Tumbleston, D. Shirvanyants, N. Ermoshkin, R. Janusziewicz, A. R. Johnson, D. Kelly, K. Chen, R. Pinschmidt, J. P. Rolland, A. Ermoshkin, E. T. Samulski, J. M. DeSimone, *Science* **2015**, *347*, 1349.