J. Mech. Phys. Solids 186 (2024) 105561

Contents lists available at ScienceDirect

Journal of the Mechanics and Physics of Solids

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

Machine learning and sequential subdomain optimization for ultrafast inverse design of 4D-printed active composite structures

Xiaohao Sun [a,1], Luxia Yu [a,1], Liang Yue [a,1], Kun Zhou [b], Frédéric Demoly [c,d], Ruike Renee Zhao [e], H. Jerry Qi [a,*]

[a] The George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[b] Singapore Centre for 3D Printing, School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore
[c] ICB UMR 6303 CNRS, Belfort-Montbeliard University of Technology, UTBM, 90010 Belfort, France
[d] Institut Universitaire de France (IUF) , Paris, France
[e] Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA

ARTICLE INFO

Keywords:
Active composites
Morphing structures
4D printing
Inverse design
Machine learning

ABSTRACT

Shape transformations of active composites (ACs) depend on the spatial distribution and active response of constituent materials. Voxel-level complex material distributions offer a vast possibility for attainable shape changes of 4D-printed ACs, while also posing a significant challenge in efficiently designing material distributions to achieve target shape changes. Here, we present an integrated machine learning (ML) and sequential subdomain optimization (SSO) approach for ultrafast inverse designs of 4D-printed AC structures. By leveraging the inherent sequential dependency, a recurrent neural network ML model and SSO are seamlessly integrated. For multiple target shapes of various complexities, ML-SSO demonstrates superior performance in optimization accuracy and speed, delivering results within second(s). When integrated with computer vision, ML-SSO also enables an ultrafast, streamlined design-fabrication paradigm based on hand-drawn targets. Furthermore, ML-SSO empowered with a splicing strategy is capable of designing diverse lengthwise voxel configurations, thus showing exceptional adaptability to intricate target shapes with different lengths without compromising high speed and accuracy. As a comparison, for the benchmark three-period shape, the finite element and evolutionary algorithm (EA) method was estimated to need 219 days for the inverse design; the ML-EA achieved the design in 54 min; the new ML-SSO with splicing strategy requires only 1.97 s. By further leveraging appropriate symmetries, the highly efficient ML-SSO is employed to design active shape changes of 4D-printed lattice structures. The new ML-SSO approach thus provides a highly efficient tool for the design of various 4D-printed, shape-morphing AC structures.

1. Introduction

Shape transformations of active composites (ACs) depend on the spatial distribution and active response of constituent materials. Multimaterial 3D/4D printing (Ge et al., 2013; Kuang et al., 2019; Cheng et al., 2022; Yue et al., 2023; Yue et al., 2023) allows for easy

* Corresponding author.
E-mail address: qih@me.gatech.edu (H.J. Qi).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.jmps.2024.105561
Received 26 November 2023; Received in revised form 26 January 2024; Accepted 29 January 2024
Available online 2 February 2024

implementation of voxel-level distribution of active materials/properties, thus permitting a vast space of active shape changes that can be achieved. To fully exploit this space, inverse design (i.e., finding the optimal material distribution to achieve a target shape) is highly desired, which is yet challenging due to the tremendous design space (Demoly et al., 2021; Sun et al., 2024). To tackle these challenges, gradient-based and gradient-free methods have been developed. For instance, the gradient-based topology optimization (TO) has been used for designing active shape changes of ACs (Maute et al., 2015; Geiss et al., 2019; Tanaka et al., 2023) or optimizing structure compliances (Zolfagharian et al., 2020). The TO, however, typically requires the complicated derivation of gradients and may encounter difficulties when the ACs involve geometric or material nonlinearity (Wang et al., 2021; Zhao and Zhang, 2022). Alternatively, the gradient-free evolutionary algorithm (EA) has also been adopted for designing active shape changes of ACs (Hamel et al., 2019; Wu et al., 2020; Athinarayanarao et al., 2023). EA is well suited for handling discrete design variables and can scale efficiently on parallel computing. However, its lack of gradient information means there is no directional guidance for the solution search, resulting in inefficiency (Sigmund, 2011). As a consequence, these EA implementations (Hamel et al., 2019; Wu et al., 2020; Athinarayanarao et al., 2023), which require numerous shape predictions through finite element (FE) methods (referred to as FE-EA), suffer from high computational cost. Although reduced-order models have been developed to replace FE for faster EA, they suffer from either relatively low accuracy (against FE) (Sossou et al., 2019) or limited design freedom (1D material distribution) (Wang et al., 2021). Accurate and efficient inverse design strategies remain to be developed.

Machine learning (ML) (Guo et al., 2021) opens up new avenues for developing fast, computationally affordable, and high-fidelity models for both forward prediction and inverse design of a variety of material responses. Existing works mainly focused on optimizing or predicting mechanical properties of materials, such as the strength and toughness of composites (Gu et al., 2018; Chen and Gu, 2019), stress and strain fields of composites (Yang et al., 2021), Poisson's ratio of auxetic metamaterials (Wilt et al., 2020), responses of soft pneumatic robots (Zolfagharian et al., 2021), among others (Roach et al., 2021; Rawat and Shen, 2018). However, there is limited work on exploiting ML for designing active shape-change responses of ACs (Sun et al., 2024). For voxelized AC beams, Zhang and Gu (2020) explored performances of multiple ML models for the forward prediction problem. Our recent work (Sun et al., 2022) demonstrated that the recurrent neural network (RNN) is particularly suited for the beam problem as it inherently preserves a sequential dependency, similar to that in beam deflection. The RNN-based ML model thus demonstrated exceptional accuracy in forward shape predictions based on material distributions. The ML was then integrated with EA (ML-EA), enabling fast inverse designs for complicated target shapes, typically in 10–60 min. Although ML-EA has significantly outperformed conventional methods such as FE-EA, it also has two major limitations. First, there remains a strong demand for faster or even real-time inverse designs for AC beams. One key limiting factor for the design speed is that EA seeks solutions in a vast design space comprising all voxels with no preferred search direction, which is inefficient. Second, an ML model, once trained, is specific to a certain beam length and voxel configuration, making it inapplicable to designs of target shapes that have different lengths (hence different voxel numbers) or finer features (such as large curvature) beyond the design space of current voxels.

This work addresses these issues by leveraging the sequential dependency of beam deflections. We first present a design approach that seamlessly combines an RNN ML model and a sequential subdomain optimization (SSO) algorithm to realize ultrafast inverse designs for target shapes of various complexities, producing results in second(s), which is hundreds or thousands of times faster than conventional ML-EA method. In addition, the sequential characteristic allows for accurate predictions for beams with fewer lengthwise voxels than that of the training data. It also permits the splicing of multiple predicted shapes, extending accuracy to beams with more lengthwise voxels. Moreover, this splicing strategy, when integrated with ML-SSO, also permits ultrafast inverse designs for a spectrum of target intricate shapes with different lengths, using lengthwise voxel configurations different from that for the ML training. Finally, the highly efficient ML-SSO is employed for the inverse design of active shape changes of 4D-printed lattice structures. Our approach thus facilitates the development of 4D printing towards intelligent and streamlined design and fabrication of various shape-morphing AC structures.

2. Models and methods

2.1. Physical problem

The physical problem is illustrated in Fig. 1a. We consider an AC beam whose left end is fixed. It is composed of N_x (length- or x-direction) $\times$ N_y (thickness- or y-direction) voxels, each assigned with an active or a passive material. The active material (encoded by "1″) expands under the actuation while the passive material (encoded by "0″) does not. The material distribution is therefore digitally encoded into an $N_y \times N_x$ binary number array, denoted by $\mathbf{M}$. Under an actuation, an active shape change is induced by the expansion mismatch, which depends on the material distribution (Fig. 1a). The shape is represented by the coordinates (x, y) sampled from certain points of voxels, forming 2D number arrays $\mathbf{x}$ and $\mathbf{y}$. Here, we focus on the voxel configuration of 24 (N_x) $\times$ 4 (N_y), which has a large design space involving 2^{96} ($\approx 7.92 \times 10^{28}$) possible material distributions. Our goal is to build an ML model to predict the mapping from $\mathbf{M}$ to ($\mathbf{x}$, $\mathbf{y}$). To achieve this, the FE simulations are performed to generate a dataset.

2.2. Finite element model

We perform FE simulations using the commercial software Abaqus/Standard (version 2018, Simulia, Providence, RI). The FE model follows our previous work (Sun et al., 2022) and is briefly summarized here. An active beam with the left edge fixed ($x = y = 0$) and under the plane strain conditions is considered. The beam has a dimension of 80 mm long $\times$ 1 mm thick and is partitioned into $N_x \times N_y$ voxels for material assignment. Both active and passive materials are modeled as nearly incompressible neo-Hookean solids with the

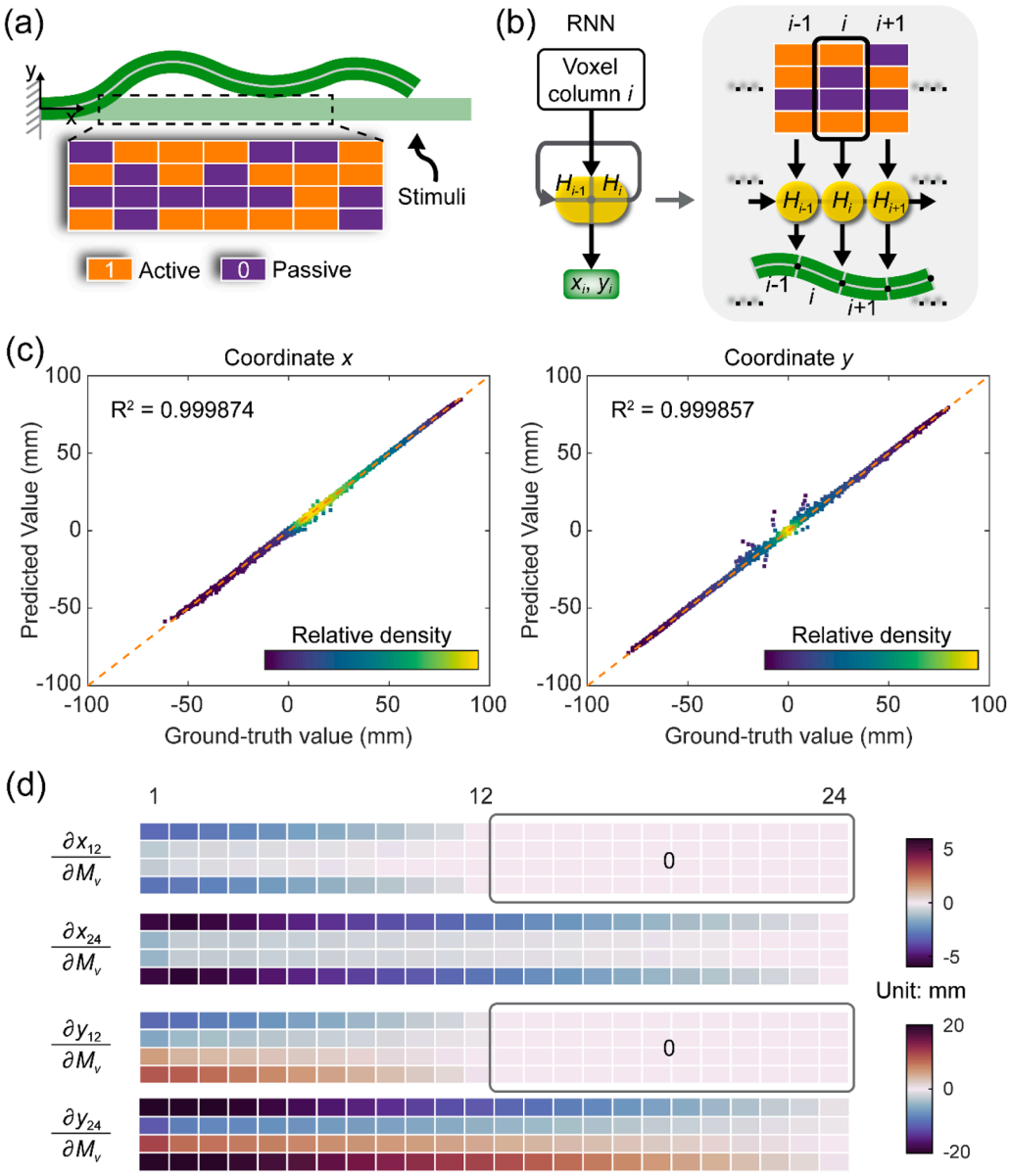

Fig. 1. Schematic illustration of the active composite (AC) beam and the sequential dependency. (a) Actuation of an AC beam with voxel-level material distribution. The two materials, encoded as "1″ (active) and "0″ (passive), have a volume expansion mismatch. (b) Architecture of an RNN, which shows a similar structure to the AC beam. (c) Density scatter plots of the ground-truth versus ML-predicted values of the coordinates x (left) and y (right) in the test dataset. The color indicates the relative point density. (d) Gradients (or sensitivities) of actuated coordinates (x, y) of the 12th and 24th voxel columns with respect to all voxels (i.e., $\partial x_i/\partial M_v$ and $\partial y_i/\partial M_v$ for i = 12 and 24, where $v = 1, 2, \ldots, 4 \times 24$ represents the voxel number, M_v represents the material encoding of voxel v), based on an all-passive (M_v=0) state. The color of an arbitrary voxel represents the gradient of x_i or y_i (unit: mm) with respect to the encoding of this voxel (unitless).

same Young's modulus but different coefficients of thermal expansion (0.001 for active and 0 for passive materials). The shape change is induced by applying a 100 °C temperature increase to the entire beam, which results in a linear strain mismatch of 0.1 between the two materials. A finite strain simulation framework is utilized (The "Nlgeom" is set to "On"). A mesh convergence study is performed and 960 × 12=11,520 hybrid plane strain (CPE4H) elements are adopted. The FE model is automatically generated and executed through a Python script. To generate a dataset, 10,800 random material distributions are created, and FE simulations are used to obtain their ground-truth actuated shapes. The generated dataset, which consists of 10,800 pairs of **M** and **(x, y)**, is then randomly split into training, validation, and test datasets with sizes of 6000, 2000, and 2800, respectively. Each of **M, x**, and **y** is restructured into sequential data: **M** becomes a sequence of N_x columns of voxels, and **x** or **y** becomes a sequence of N_x coordinate values (each sampled from one column). In general, sampling points can be arbitrarily chosen from available mesh points. Here, we choose the sampling points to be the mid-axis, rightmost mesh point of each column of voxels.

2.3. RNN-based ML model for forward predictions

With the dataset generated by FE simulations, an RNN-based ML model is constructed to predict actuated shapes for given material distributions (Fig. 1b). RNN can use the "past" information to predict the "current" response and thus is highly capable of learning sequential data. It is noted that the deflection of the AC beam has a similar sequential characteristic, i.e., the displacement of any points only depends on the points to their left (or before them); the displacement of the points to their right (or after them) would not affect their displacement. This feature inspires us to use RNN.

For each step, RNN receives a single column of voxel encodings and produces the coordinates of the column (sampled on the mid-axis of the beam). By repeating this process, RNN sequentially processes all voxel columns to predict the shape of the entire beam (Fig. 1b). More details on the construction of the RNN model are provided in Appendix A.1.

Two RNN-based ML models are trained separately, one for predicting coordinate x and the other for coordinate y, to better identify their respective errors. Let F_x and F_y denote the built models for x and y, respectively, we have

$$\mathbf{x} = F_x(\theta_x; \mathbf{M}), \mathbf{y} = F_y\left(\theta_y; \mathbf{M}\right), \tag{1}$$

where **x** and **y** are predicted coordinate vectors whose components, x_i and y_i, $i \in [1, N_x]$, are coordinates of sampling points, respectively. θ_x and θ_y denote the learnable parameter sets for F_x and F_y, respectively. The loss function is defined as the half-mean-squared-error between the predicted coordinates (x_i or y_i) and true coordinates (x_i^{true} or y_i^{true}) for a specific material distribution, i.e.,

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

for coordinates x and y, respectively. The training of ML models is to find the optimal θ_x and θ_y that minimize the loss values on the training set, i.e., $\underset{\theta_x}{minLoss}(x)$and $\underset{\theta_y}{minLoss}(y)$. The adaptive moment estimation (Adam) (Kingma and Ba, 2014) optimizer is used to train the network. We perform a systematic study on the hyperparameters for the training and network architectures, which is detailed in Appendix A.2.

2.4. Performance of the ML model

As shown in our previous study (Sun et al., 2022), the RNN-based ML model demonstrates high accuracy for cases with different numbers of voxels. In this work, we adopt the ML model trained on 24 (N_x) × 4 (N_y) voxels. Notably, the coordinates (x, y) are sampled from the mid-axis, rightmost mesh point of each voxel column to represent a shape, which is different from the middle point as used in our previous work (Fig. 1b). This choice of sampling points has two merits. First, it achieves higher prediction accuracy with R^2 = 0.99987 for x and 0.99986 for y, as shown in regression (or density scatter) plots of the ground-truth versus ML-predicted coordinates of the test dataset (Fig. 1c). The increased accuracy is attributed to the better sequential characteristic of the column rightmost points. Second, the new sampling points enable the ultrafast SSO optimizations as will be introduced in Section 2.5. Although the model achieves very high accuracy, a few datapoints deviate from the regression line (Fig. 1c). The analysis of prediction errors shows that the data with large errors are only a small portion (<0.36 %) of the test set (Fig. A.4 and associated text in Appendix A.3). Even for these cases, the accuracy is very high near the beam origin and over most of the beam length, and the large errors all happen near the beam end. In addition to the accuracy, the ML model allows for ultrafast prediction speed due to its parallel computing capability on GPU (see Appendix A.4).

As mentioned above, since the left end of the actuated beam is fixed, the coordinate of a voxel column is dependent on the voxels in the current column and the left of the current column. Similarly, the architecture of RNN determines that the "current" output is dependent on the "current" and "past" inputs. To illustrate this feature, we display the gradients (or sensitivities) of RNN-predicted coordinates (x, y) of the 12th and 24th voxel columns with respect to all voxels (i.e., $\partial x_i/\partial M_v$ and $\partial y_i/\partial M_v$ for i = 12 and 24, where $v = 1, 2, \ldots, 4 \times 24$ represents the voxel number, M_v represents the material encoding of voxel v), based on a state where all voxels are initially passive materials (M_v=0) (Fig. 1d). The gradients are calculated through automatic differentiation as described in Appendix A.5. We see that (x_{12}, y_{12}) depends on the first 12 voxel columns only, while (x_{24}, y_{24}) depends on all the voxels. The results evidently showcase that the RNN inherently preserves the sequential data dependency of the beam deflection.

Such a similarity in sequential dependency allows RNN to achieve superior accuracy over other network architectures such as neural network (NN) and convolutional neural network (CNN), as demonstrated in Fig. A.3. Note that RNN uses the "temporally" invariant weights across different steps and thus is well suited to learn a universal relationship between two consecutive voxel columns. In contrast, CNN uses spatially invariant weights (convolutional kernels) to recognize spatial local features, which intuitively have a highly complex mapping to the coordinates of large-deflection beams. In addition, CNN likely needs a large reception field to learn the relation between the voxels near the origin and the coordinates near the end, which leads to inefficiency. For example, each coordinate would depend on the global voxel information, and thus it may be easily disturbed by irrelevant voxels.

2.5. ML-SSO approach for ultrafast inverse design

Although the ML model achieves high accuracy in the forward prediction, applying it to the inverse map (from shape to design) is still challenging due to the "one-to-many" characteristic (i.e., one actuated shape could be achieved by distinct designs)[7]. This is particularly the case considering the large design space and the high-dimensional data for shape representation. To address this challenge, a straightforward approach is to combine ML with optimization algorithms such as EA (Sun et al., 2022). However, as a gradient-free stochastic search algorithm, EA seeks solutions in a vast design space comprising all voxels with no preferred search direction, which is inefficient.

In this work, we present an ML-empowered SSO approach that can further leverage the inherent sequential characteristic. Instead of directly optimizing the entire domain, this approach sequentially optimizes subdomains, composed of N_{sub} columns, from left to right (Fig. 2). In each optimization step, SSO receives the coordinates of N_{sub} target points (highlighted red "o") and optimizes the material distribution in the corresponding subdomain (highlighted yellow domain) such that the RNN-predicted coordinates of actuated points (highlighted blue "+") match with the targets. Once a subdomain, e.g., (i-1), is optimized, the target points and design domain move one step forward for the next subdomain, i, as seen by the one-step sliding of the highlighting window in Fig. 2. Due to the sequential dependency, the optimization of subdomain i would not affect the optimized preceding subdomains (those to the left of subdomain i) except the overlapping domain of (i-1) and i (see the rightmost panel of Fig. 2). Here, we choose N_{sub} to range from 1 to 3. When N_{sub}=1, there is no overlap between two consecutive subdomains. As will be detailed later, a larger N_{sub} can improve the optimization accuracy. For the optimization of individual subdomains, we employ a brute-force approach, calculating the subdomain shapes of all possible designs through RNN and retaining the optimal one. When evaluating these shapes, RNN receives the subdomain inputs (with N_{sub} columns) to produce their corresponding coordinates. The change of N_{sub} does not need the retraining of RNN. As N_{sub} increases, the number of potential designs ($=2^{Ny\times Nsub}$) would increase exponentially, and therefore a cap on N_{sub} is chosen to be 3. This ensures the local optimality of the found solution while not requiring many shape evaluations as the chosen N_{sub} is small. For larger N_{sub}, the subdomain EA or other stochastic search algorithms may be used to reduce computational cost. The loss function L_{sub} for each subdomain is defined as

$$L_{sub} = \sqrt{\frac{1}{N_{sub}}\sum_{j=1}^{N_{sub}}\left[\left(x_{sub,j}-\widehat{x}_{sub,j}\right)^2+\left(y_{sub,j}-\widehat{y}_{sub,j}\right)^2\right]} \tag{3}$$

where ($x_{sub,j}$, $y_{sub,j}$) are the ML-predicted sampled point coordinates for the j-th column of the considered subdomain, $(\widehat{x}_{sub,\,j}, \widehat{y}_{sub,j})$ are the corresponding target coordinates. Compared to the ML-EA on the entire domain, ML-SSO reduces computational cost in two aspects. First, the design space of a single subdomain is much smaller than the whole domain, thus drastically reducing the number of potential design evaluations required. For instance, for N_{sub} = 2, the brute-force method assesses 256 ($=2^{Ny\times Nsub}=2^8$) potential subdomain designs per step, totaling 5888 ($=256\times 23$, where $23=(N_x-N_{sub}+1)$ is the number of subdomains to optimize) evaluations, which is much smaller than the number (typically 75,000–375,000 (Sun et al., 2022)) of global ML-EA. Second, these are the number of potential designs that need to be evaluated. In ML-SSO, the evaluation of each candidate design (comprising N_{sub} columns only) is also more efficient. This is because, during each optimization step, RNN permits only processing the current subdomain and predicting its coordinates ($x_{sub,j}$, $y_{sub,j}$), $j\in[1, N_{sub}]$, based on the stored hidden state, thereby bypassing the need to evaluate the preceding or

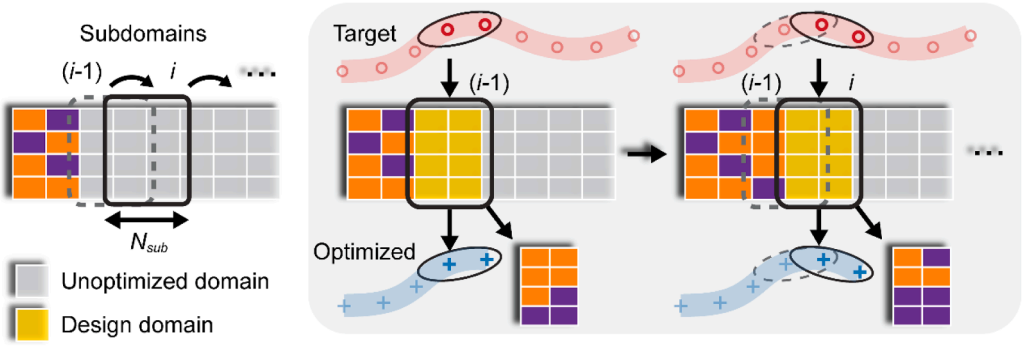

Fig. 2. Schematic of the ML-SSO approach.

subsequent subdomains. This further significantly reduces computational costs compared to ML-EA which demands a full-domain shape evaluation for each candidate design.

3. Results and discussions

3.1. Performance of ML-SSO on numerically generated target shapes

We now consider multiple numerically generated target shapes of different complexities to evaluate the performance of ML-SSO with different N_{sub} values. In this sub-section, we focus on the case with 24 × 4 voxels. The first two target shapes, the one-period and two-period sinusoidal shapes, are optimized using N_{sub} = 1. To provide a more intuitive understanding of ML-SSO, in Figs. 3a and 3b, we show the optimized and target shapes with symbols of varying colors to visualize the optimization sequence. Each optimized point ("+") is the one closest to the corresponding target point ("o", same color) among all possible designs for the specific subdomain. For both two targets, excellent agreements between the optimized shapes and the targets are achieved. Although some discrepancies can be observed, e.g., in the middle of the two-period shape where the curvature is relatively large, the subsequent optimized points can still accurately capture the target shape (Fig. 3b), demonstrating the high capability of our approach.

Next, two more complicated targets are considered: the three-period shape and the half-butterfly shape. As shown in Fig. 3c and d, the sensitivity of optimized shapes to the N_{sub} is studied. For both targets, N_{sub}=1 leads to notable discrepancies between the optimized and target shapes. With the increased N_{sub}, the optimization accuracy is significantly improved. This is because ML-SSO operates on a greedy strategy, selecting the best shape at each step. Such an approach might lead to a short-sighted problem for regions with sharp turns, missing the target of the subsequent subdomain in favor of the best match for the current one. Increasing N_{sub} effectively broadens the optimization foresight and can, to some extent, mitigate this problem, although it concurrently escalates the computational cost. To gain more insights, we show in Fig. 3c and d the optimization sequence for the cases with N_{sub}=2, where each voxel

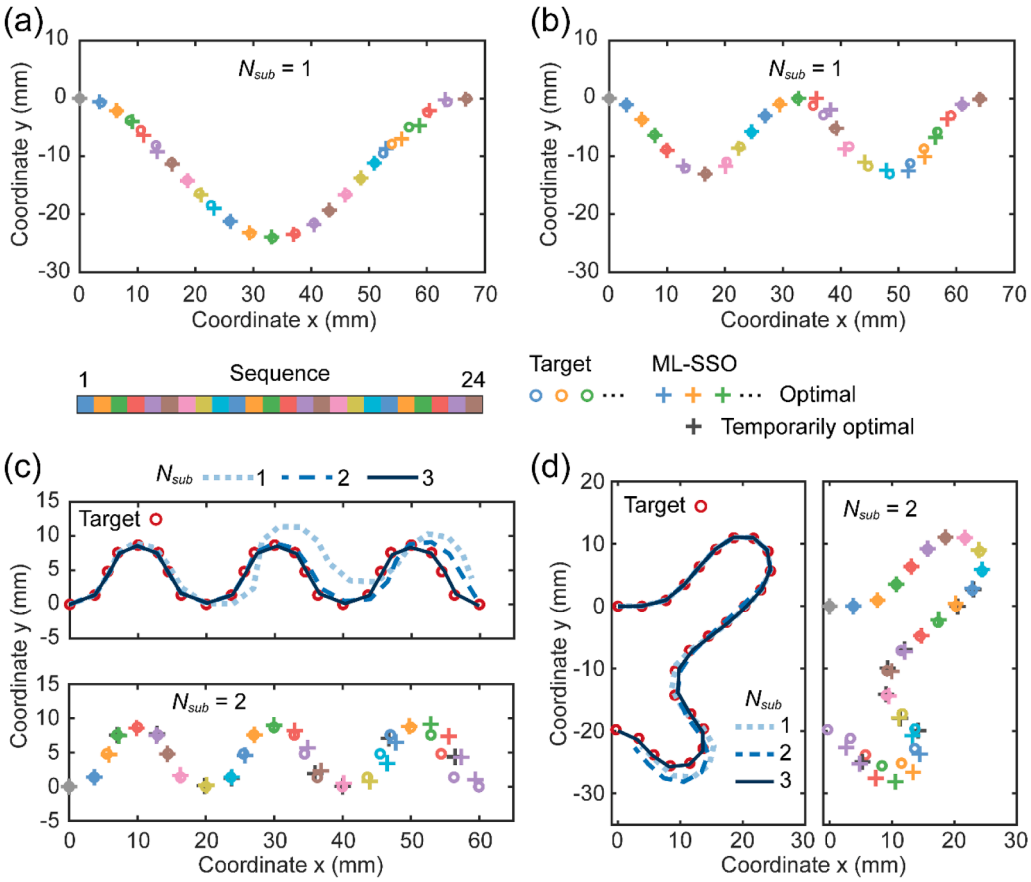

Fig. 3. Demonstration of ML-SSO on inverse designs of numerically generated target shapes. (a) One-period target and ML-SSO optimized shape with N_{sub}=1. (b) Two-period target and ML-SSO optimized shape with N_{sub}=1. (c) Three-period target, ML-SSO optimized shape with different N_{sub} values, and illustration of optimization sequence with N_{sub}=2. (d) Half-butterfly target, ML-SSO optimized shape with different N_{sub} values, and illustration of optimization sequence with N_{sub}=2. In the plots with optimization sequence, the last N_{sub} points are optimized in one step and thus have the same color. All ML-SSO designs are based on N_x=24.

column (except the first and last ones) is involved in two consecutive subdomains and thus optimized twice (see, for example, the overlapping domain of (i-1) and i in the rightmost panel of Fig. 2). The temporarily optimal point denotes the one achieved in the first optimization of its involved subdomain and is depicted in dark gray "+" symbols. The discrepancy between the temporarily optimal point and the optimal point often appears in regions with sharp turns, manifesting the short-sighted problem.

We further quantitively compare the performance of ML-SSO with different N_{sub} values on these four targets in terms of accuracy (Fig. 4a) and time cost (Fig. 4b). The ML-EA optimization results are already available (Sun et al., 2022) and also compared here. The accuracy is quantified by the root-mean-squared errors (RMSEs)

$$\mathrm{RMSE} = \sqrt{\frac{1}{2N_x}\sum_{i=1}^{N_x}\left[(x_i - \widehat{x}_i)^2 + (y_i - \widehat{y}_i)^2\right]} \tag{4}$$

between the optimized coordinates (x_i, y_i) and target coordinates $(\widehat{x}_i, \widehat{y}_i)$ for all N_x sampling points. As shown in Fig. 4, for the relatively simple one-period and two-period shapes, ML-SSO with N_{sub}=1 which only takes 0.23 s achieves similar RMSEs with that of ML-EA which takes 660 s (or 11 min). Using the number of shape evaluations needed in ML-EA, the FE-EA approach was estimated to need 44 days for these two targets, as discussed in our previous work (Sun et al., 2022). For the most complex three-period shape, ML-SSO with N_{sub}=3 takes 11.8 s and achieves much better RMSEs than that of ML-EA, which takes 3240 s (or 54 min); the FE-EA approach was estimated to need 219 days (Sun et al., 2022). It is worth noting that a large time increase can be observed as N_{sub} increases, which is due to the exponentially growing brute-force shape evaluations. Interestingly, the time increase factor is about 4 when the N_{sub} increases from 1 to 2 which corresponds to a 16-fold increase in the number of shape evaluations. This acceleration in optimization is due to the faster ML prediction enabled by its parallel computing capability on GPU (see Appendix A.4). In short, ML-SSO outperforms or matches ML-EA in optimization accuracy while being significantly (at least two orders of magnitude) faster. This demonstrates the very high efficacy of our approach in solving inverse material design problems for complicated target shapes, which is attributed to the seamless integration of RNN-ML and SSO that fully leverages sequential characteristics for highly efficient optimal design searches.

3.2. ML-SSO enabled, streamlined 4D printing design and fabrication based on hand-drawn lines

Next, we apply the ultrafast ML-SSO approach to hand-drawn target shapes to enable a streamlined design-fabrication paradigm for 4D printing. In general, these target shapes are more challenging to optimize, because the discrete target points are uniformly sampled from the identified shape, which could have inappropriate spacing and thus produce difficulties for the optimization. Following our previous work (Sun et al., 2022), computer vision (CV) algorithms are employed for the automatic identification of target shapes from hand-drawn lines. The ML-SSO is then utilized for the inverse design, and the optimized material distributions can be readily

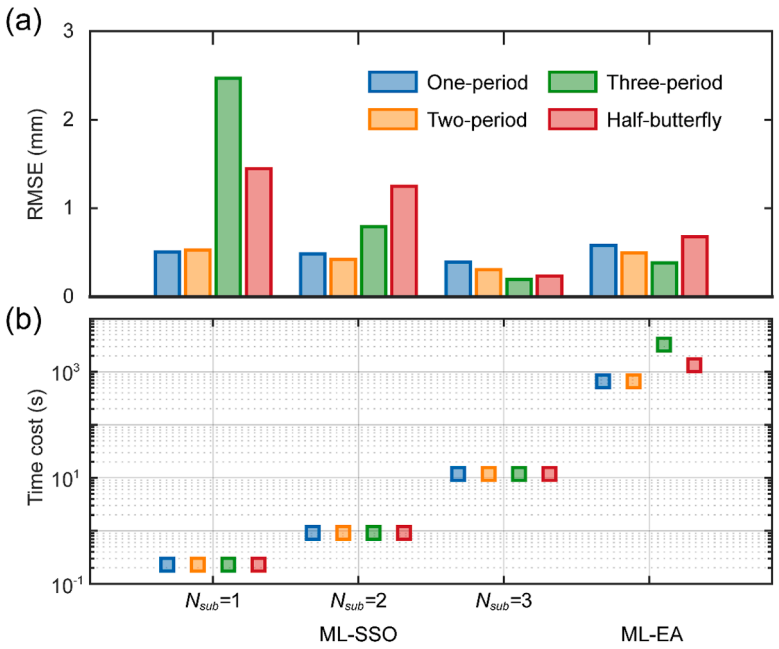

Fig. 4. (a) RMSEs of optimized shapes by ML-SSO (with different N_{sub} values) and ML-EA for the four target shapes considered in Fig. 3. The RMSE is calculated using Eq.(2). (b) Time cost of the corresponding optimizations, which are run using the Intel Core i9-10900 CPU and NVIDIA Quadro P620 GPU.

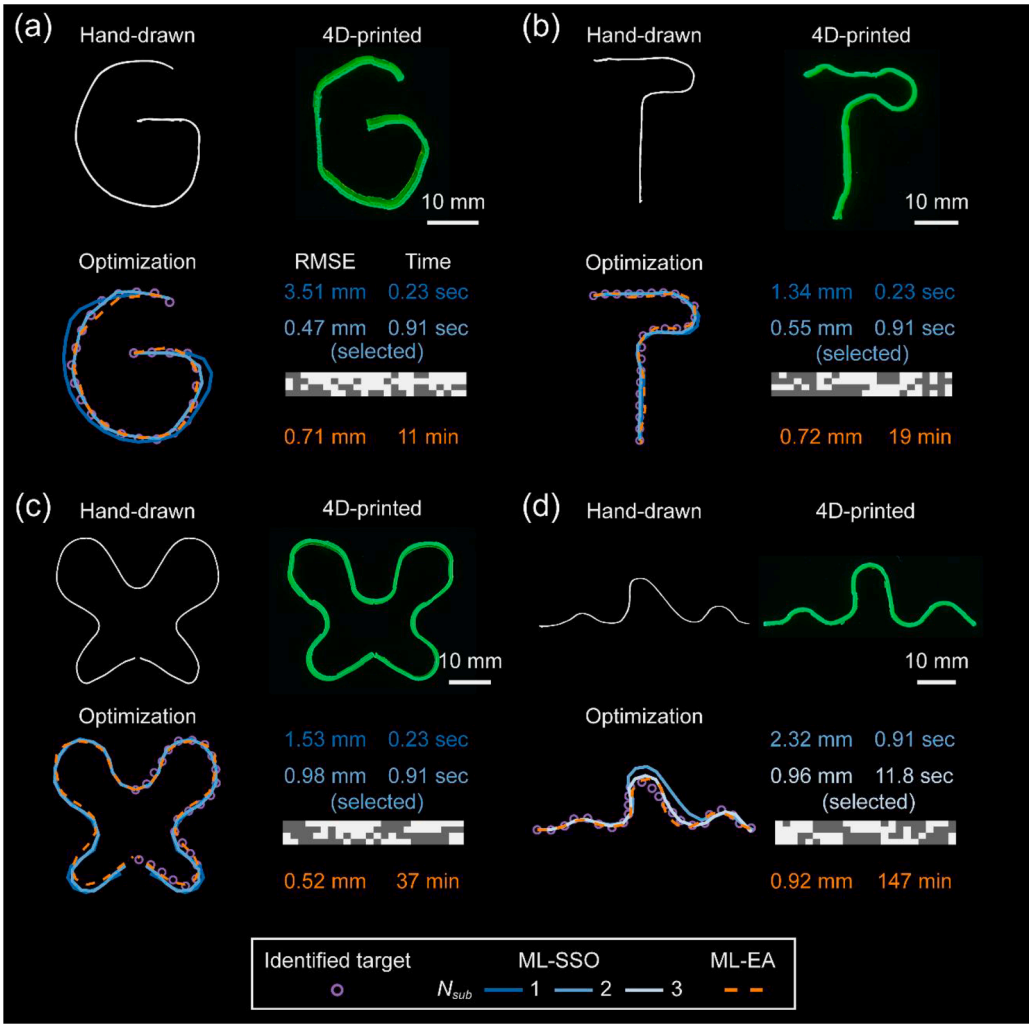

Fig. 5. Ultrafast ML-SSO enabled streamlined design and fabrication for 4D printing based on hand-drawn lines. Four hand-drawn targets are considered: (a) "G", (b) "T", (c) half-butterfly (the other half is also displayed), and (d) mountain. Shown in each panel includes the hand-drawn line (top left), optimized shapes by ML-SSO (with different N_{sub} values) and ML-EA (bottom left), the corresponding RMSE and time cost as well as the optimal ML-SSO design selected (RMSE $<$ 1 mm, shown in grayscale) (bottom right), and the 4D-printed, actuated shape (top right).

4D-printed using the grayscale digital light processing (g-DLP)[2, 4] technique. The material system is different from that of our previous work (Sun et al., 2022). More details on the printing method, material, and actuation mechanism are provided in Appendix B.

Fig. 5 shows the ML-SSO design and fabrication results for four different target shapes. The ML-EA optimized designs are available (Sun et al., 2022) and also presented here for the evaluation of the ML-SSO performance. For each target shape, shown in four sections are the hand-drawn line (top left), the optimized shapes by ML-SSO (with different N_{sub} values, solid lines) and ML-EA (dashed lines) compared against the targets (symbols) (bottom left), the corresponding performance in terms of RMSE and time cost as well as the optimal ML-SSO design selected (RMSE $<$ 1 mm, shown in grayscale) (bottom right), and the experimentally 4D-printed, actuated shape (top right). All printed strips are initially flat and only the actuated shapes are shown. For all the targets, the optimized shapes by both ML-SSO and ML-EA agree well with the target shapes, whilst ML-SSO is much faster than ML-EA. The optimal ML-SSO designs are finally converted into grayscale slices for 4D printing through g-DLP, and the printed strips transform into target shapes upon actuation. These results demonstrate an ultrafast, streamlined design-fabrication paradigm for 4D printing.

3.3. Active beams with varying numbers of lengthwise voxels: forward predictions

The preceding sections demonstrate ultrafast inverse designs enabled by ML-SSO for both numerically generated and hand-drawn target shapes. However, these designs have been constrained to fixed beam length (80 mm) and voxel configuration (24 $\times$ 4 voxels) on which the ML model is trained, implying a relatively limited design space. For example, a configuration of N_x=24 lengthwise voxels might struggle to capture complicated target shapes, such as a six-period shape, which would be very challenging for N_x=24 as it would

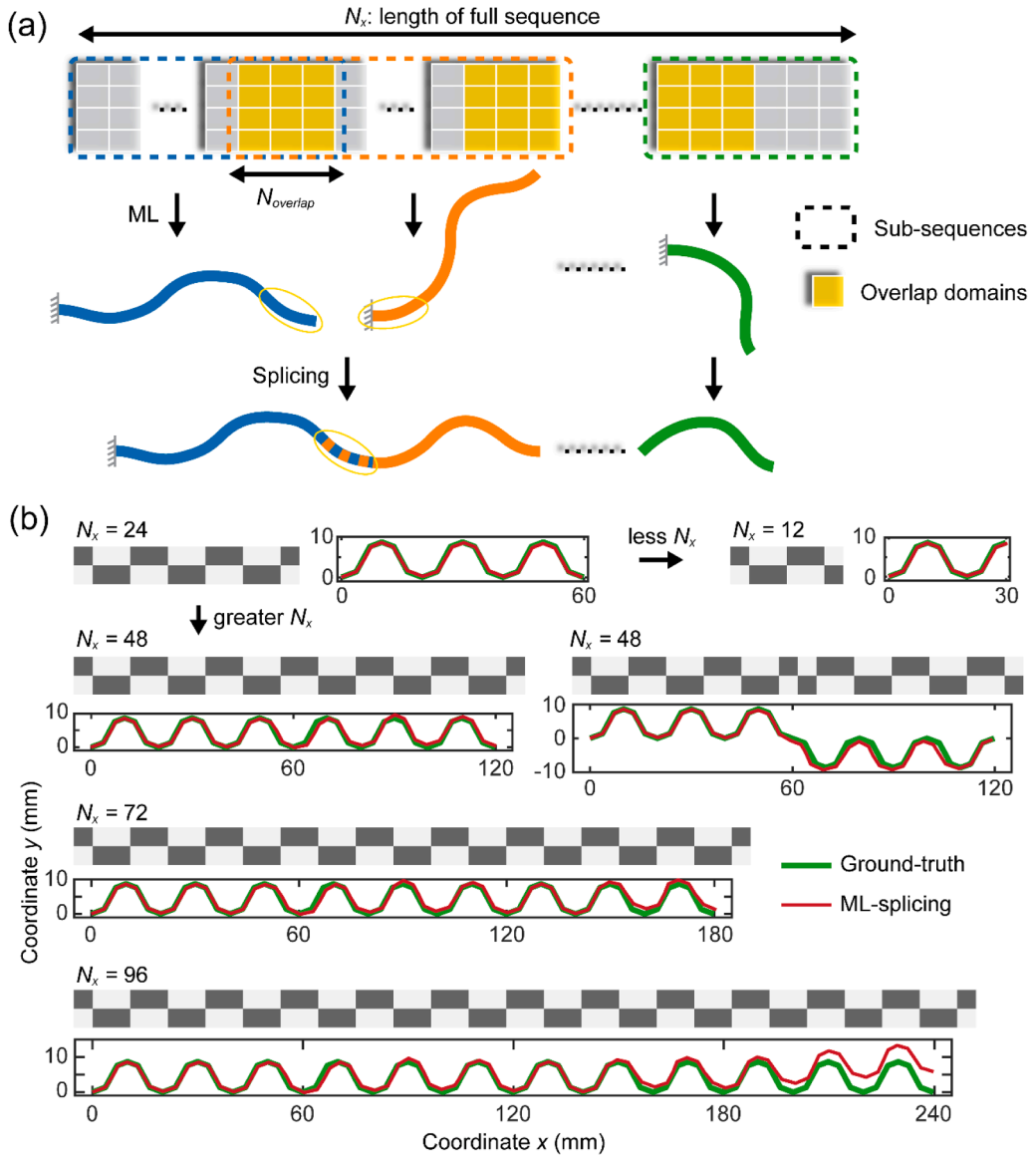

Fig. 6. Demonstration of ML-splicing on forward prediction problems of beam structures with different N_x values. (a) Schematic of ML-splicing. (b) Prediction results for benchmark input sequences of different N_x values.

require four voxels in the length direction to capture one period. Therefore, more lengthwise voxels ($N_x > 24$) are needed to tackle this design problem.

To further enhance design capabilities, we aim for the ML-based method to address both forward and inverse problems for beam structures with varying lengths or N_x values. Here, we fix the voxel number N_y=4 in thickness and the voxel size. Thus, the sequential (column-by-column) prediction ability of the RNN-ML allows it to be directly applied to structures with different N_x values and hence different lengths. For the case N_x>24, we propose an ML-based splicing method, with no need for retraining the model, for forward predictions with different N_x values.

As shown in Fig. 6a, an input sequence (referring to a material distribution) of any length N_x can be divided into multiple sub-sequences with overlaps, each with a length of 24, except for the last subsequence which can be shorter than 24. We use ML to predict the shapes of all subsequences, and then concatenate them in order, to obtain the full-sequence shape. The predicted shapes of all subsequences are initially based on a fixed boundary condition on the left end, which we refer to as being in their local spaces, while the shape of the first subsequence is in the real space. We sequentially transform each subsequence shape into the real space by matching the coordinates in the overlapping regions between two consecutive subsequences. Here, we use the length of the overlapping region to be $N_{overlap}$=6. Note that the correct local concatenation angle cannot be determined if there is no overlapping region.

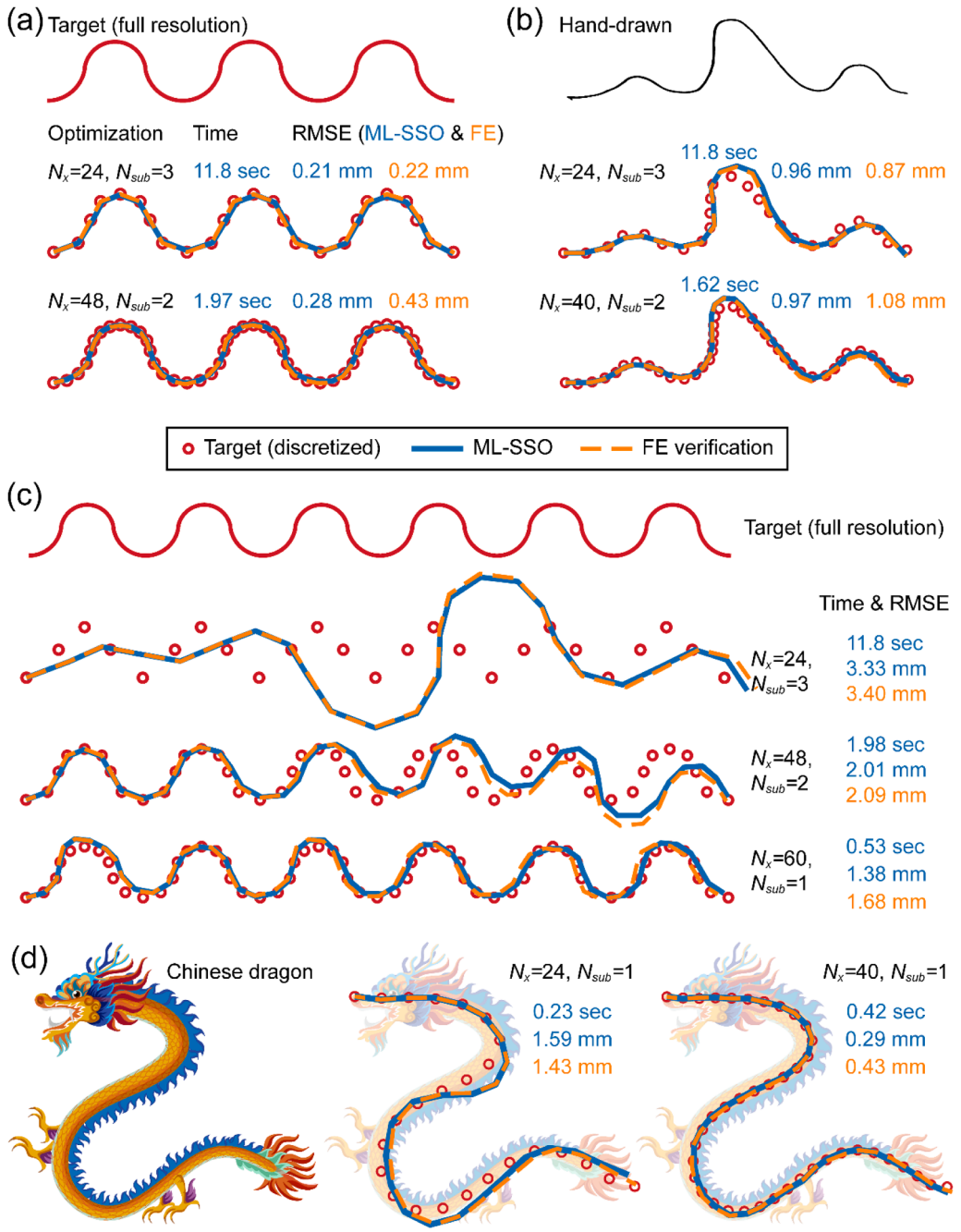

Fig. 7. Demonstration of the splicing-empowered ML-SSO on inverse design problems using different N_x (numbers lengthwise voxels) or numbers of sampling points. Multiple target shapes are considered, including (a) the three-period shape, (b) the hand-drawn mountain shape, (c) a six-period shape, and (d) a "Chinese dragon" profile. Image by brgfx on Freepik (https://www.freepik.com/free-vector/chinese-dragon-flying-clouds-isolated-white-background_18054261.htm).

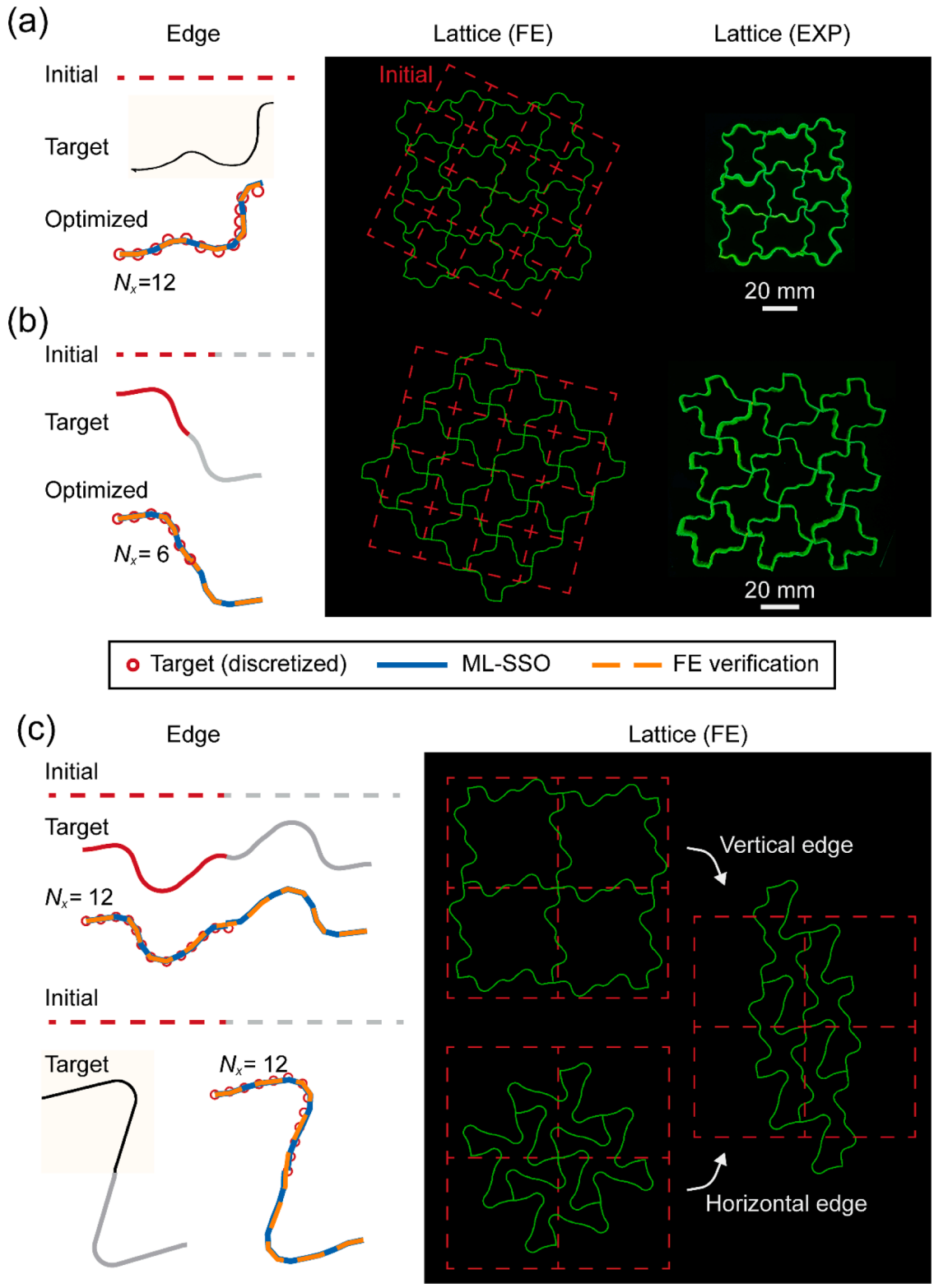

Fig. 8. Applications of ML-SSO to rapid designs for active lattice structures. (a) Arbitrary edge design with half of the hand-drawn mountain as a target. (b) Anti-symmetric edge design with a simple wavy target. (c) Anti-symmetric edge design with two more complex target shapes. In (a-c), the dashed lines represent the initial shapes of edges or lattices before actuation.

Next, we perform forward predictions on multiple benchmark shapes of different lengths, including N_x=12, 24, 48, 72, and 96 (Fig. 6b). The input sequences with $N_x \leq 24$ are directly predicted using the RNN-ML without accuracy degradation. The input sequences with $N_x > 24$ are obtained by concatenating multiple sequences of the three-period shape and predicted using the ML-splicing strategy. Directly applying ML to such long sequences would yield less accurate results (see Fig. A.6 and associated text in Appendix A.6). By using the ML-splicing strategy, we can achieve very high prediction accuracy. Note that the ML prediction errors will inevitably accumulate with increasing N_x and numbers of splicing, as even a small prediction error in the subsequence could be retained in the shape transformation process. Nonetheless, the ML-splicing predictions remain accurate for N_x up to 72, three times N_x=24; a noticeable discrepancy only arises at N_x=96. Our strategy thus demonstrates excellent robustness significantly (three times) beyond the sequence length (N_x=24) of the ML training set.

3.4. Active beams with varying numbers of lengthwise voxels: inverse designs

Next, we combine ML-SSO with the splicing strategy for inverse design problems. The splicing strategy works in a slightly different way from the forward prediction. In the forward prediction, each subsequence shape is transformed into the real space to infer the correct shape. In the inverse design, instead of transforming the shapes of potential designs into the real space for comparisons against the target, we transform the target shape (which is initially in the real space) into the local space for optimization. This enables the seamless use of ML-SSO for each subsequence, bypassing unnecessary shape transformations and improving the optimization speed. We use multiple complicated target shapes (i.e., involving relatively sharp turns when represented using N_x=24 sampling points) to demonstrate the capability of our splicing-empowered ML-SSO. Fig. 7 shows the optimization results for these target shapes obtained with different N_x values. The optimized shapes together with the corresponding RMSEs and time cost are presented. FE simulations for the optimized designs are also performed, and the FE-predicted shapes and their RMSEs are provided. The first two targets are the most complex target shapes considered so far: the three-period shape (Fig. 7a) and the hand-drawn mountain (Fig. 7b). With the original N_x=24, N_{sub}=2 is not sufficient to accurately capture the two targets (Fig. 3c and 5d); using N_{sub}=3 significantly increases the accuracy (with RMSEs of 0.21 mm and 0.96 mm) but also takes longer optimization time (11.8 s) (Fig. 7a and 7b). By using greater N_x values (48 and 40 respectively), which offer more sampling points, these complex targets with sharp turns can be rapidly captured with N_{sub}=2, achieving similar accuracies (with RMSEs of 0.28 mm and 0.97 mm) in much shorter times (1.97 s and 1.62 s) (Fig. 7a and 7b). To put these results in perspective, for the three-period shape, the FE-EA approach was estimated to need 219 days for the inverse design, the ML-EA approach obtained the design in 54 min, and our new approach requires merely 1.97 s.

To further evaluate the capability of our approach, we consider a more complex six-period target shape obtained by chaining two three-period shapes (Fig. 7c). The original N_x=24 fails to capture this target (RMSE=3.33 mm) even with N_{sub}=3 (which takes 11.8 s), and therefore greater N_x values are used. We then use N_x=48 with N_{sub}=2 and achieve an improved accuracy (RMSE=2.01 mm) in a much shorter time (1.98 s), but an evident discrepancy can still be observed. By further increasing N_x=60, even a smaller N_{sub}=1 successfully yields a high-precision optimized shape (RMSE=1.38 mm) in only 0.53 s (Fig. 7c). As the last example, we take the profile of a "Chinese dragon" art as the target shape and again achieve excellent optimization results (RMSE=0.29 mm) very rapidly (0.42 s with N_x=40 and N_{sub}=1) (Fig. 7d).

These results show that although we train the ML model on a certain voxel configuration, the splicing strategy allows for ultrafast and accurate forward predictions and inverse designs for structures with variable N_x values and lengths by leveraging the inherent sequential characteristics.

3.5. Applications to rapid designs of 4D-printed lattice structures

The inverse designs demonstrated so far have been constrained to the AC beam structures. However, the voxel-level, ultrafast inverse shape-change designs for more complex structures such as lattice structures are highly desired, which could have broader applications but are also more challenging due to the increased number of involving voxels in an entire lattice. In this sub-section, we further show that by leveraging appropriate symmetries, the ML-SSO can be applied in designing the active shape transformation of 4D-printed lattice structures. We focus on a periodic square lattice and optimize the material distribution on its edges so that they achieve target shapes under actuation. The design strategy can be also applied to other lattice structures, such as triangular and hexagonal lattices.

First, we consider an arbitrary target edge shape. The material distribution is designed to satisfy 4-fold rotational symmetry about each joint to prevent mismatches in structural deformation. Due to the size limit of the printing stage, we use half of the hand-drawn mountain as a target (N_x=12). We adapt its optimized design for the lattice structure, validate the design through FE simulations, and then 4D print the lattice structure (Fig. 8a). The FE results show that all edges of the lattice transform to the desired shape. In experiments, we only print individual square lattices due to the size limit. These printed lattices, initially square, transform into the target shape upon actuation. When tessellated, the structure matches with the FE result. The tessellation is used solely for improved visualization and does not affect the actuated shape.

Note that for an arbitrary target shape considered above, four (2×2) lattice cells form a periodic unit. Next, we consider anti-symmetric target edge shapes, which are common in lattice structures, such that a single lattice cell would be periodic. In this case, a translational symmetry in the material distribution is satisfied, ensuring structural compatibility. Due to the anti-symmetry of the edge, the design is carried out on half of the domain. For a simple wavy target (N_x=6 on its half), we perform the inverse design and 4D printing. The initially square lattices, when actuated and tessellated, agree excellently with the FE predictions (Fig. 8b). Furthermore, we consider two more complex target shapes (N_x=12 on the half) and use them for multiple lattice structures with identical target

shapes for all edges or distinct targets for horizontal and vertical edges. For all lattices, excellent agreement is achieved among the target, the optimized shape, and the actuated edge shapes of the lattice obtained from FE simulations (Fig. 8c). These results demonstrate the high efficacy and flexibility of our design strategy for lattice structures.

It is worth noting that a translational symmetric material distribution can be used to ensure structural compatibility as long as the target edge shape has equal slopes at two ends, without necessarily being antisymmetric. Additionally, by employing appropriate target shapes for horizontal and vertical edges, our approach can be readily used for designing lattices with target macroscopic deformations, such as horizontal and vertical normal strains, shear strains, and/or rigid rotations. Moreover, our design strategy is also readily applicable to other lattice structures, such as triangular and hexagonal lattices. These results and discussions highlight the high flexibility of our approach in designing the active shape change of lattice structures.

3.6. Discussions and potential improvements

Although ML-SSO achieves high efficiency, there is still potential for further improvements, which are worthy of future investigation. First, the cap on the N_{sub} is chosen to be 3, as further increasing it would significantly increase the time cost due to the brute-force calculation. In this case, the EA or other gradient-free algorithms may be used for the subdomain optimization to enable the use of larger N_{sub} and enhance the design capability. Second, nested or adaptive optimizations may be used. For example, we may obtain an initial design using a smaller N_{sub}=1 and then refine the design at difficult regions (e.g., with a sharp kink) using larger N_{sub} values. In addition, using adaptive N_{sub} values for regions with different complexities (e.g., curvatures) could potentially improve the design accuracy and speed. Third, pre-calculating all possible shapes for a certain subdomain size ($N_{sub\text{-}pre}$) and later searching solutions in a look-up table fashion may save the time of brute-force calculations. Since the pre-calculated shapes are based on a fixed boundary condition on its left end, a splicing strategy is needed to properly convert these shapes to the correct ones, which thus requires $N_{sub\text{-}pre}$ to be larger than N_{sub} (i.e., the size of the subdomain to be optimized in SSO).

In addition, the ML model performance may also be further improved. For example, the large prediction errors mostly happen at the end of the beam (see Fig. A.4 in Appendix A.3). Based on this fact, the model performance may be improved by modifying the training loss, such as weighting the prediction errors near the end of the beam more than those near the origin (fixed end). In addition, as we train two separate networks for x and y, one single RNN for two coordinates may be trained, which could further accelerate the forward prediction and thus the inverse design. In this case, the loss weights for the two coordinates could be optimized to improve the accuracy. These potential improvements will be explored in the future.

Even with high accuracy, the ML predictions always have errors, which, in theory, could affect the optimization accuracy. However, the probability of a significantly inaccurate prediction is very low and it is almost negligible and thus not encountered in our study. This is expected since the data with significant errors are only a small portion (<0.36 %) of the test set (see Fig. A.4 and associated text in Appendix A.3). Although the probability of such a situation is low, the following measures may be taken to mitigate its impact or prevent its occurrence. First, the FE simulations should always be performed to verify the optimized designs before they are used for 4D printing. Second, one potential improvement is to add a FE verification step in the optimization. For example, one can use FE simulations to re-sort top-performing candidate solutions and select the best solution based on the FE results. A similar idea was recently proposed by Li et al. (2023). This potential improvement will be explored in our future work.

Moreover, our ML-SSO is focused on the design of 1D AC which leverages the remarkable efficiency of RNN for cantilever-like structures. However, the efficient design of higher-dimensional AC structures with 3D deformations (such as active sheets with out-of-plane actuation) could have broader applications, which is highly desired but also more challenging due to the higher-dimensional deformation behavior and significantly increased design space. The ML approach as well as the SSO and the splicing strategy could be potentially expanded to such problems, where the exploration and selection of appropriate network architectures will be important. This will be explored in our future work.

4. Conclusions

We present an approach for ultrafast inverse design of 4D-printed AC beams by combining ML and SSO. An RNN ML model is rapid and accurate in forward shape predictions based on the material distributions. We then integrate ML with SSO for the inverse design of material distributions based on the target shapes. For multiple target shapes of different complexities, ML-SSO outperforms or is comparable to ML-EA in optimization accuracy while being two to three orders of magnitude faster, delivering results in mere second (s). The CV-integrated ML-SSO then demonstrates an ultrafast, streamlined design-fabrication paradigm based on hand-drawn targets. Furthermore, although the ML model is trained on a fixed voxel configuration (N_x=24, N_y=4), we present a splicing strategy that achieves accurate shape predictions for beams with varying numbers (N_x) of lengthwise voxels with no need for retraining the ML model. This strategy, when integrated with ML-SSO, enhances adaptability for ultrafast inverse designs, accommodating target shapes of diverse complexities or lengths. For highly complex target shapes, the splicing-empowered ML-SSO, simply with N_x>24, proves to be more robust and rapid than the regular ML-SSO (which requires N_x=24), achieving better or similar optimization results while reducing time cost from 11.8 s to 1.6–2 s. As a comparison, for the benchmark three-period shape, the FE-EA method was estimated to need 219 days for the inverse design; the ML-EA achieved the design in 54 min; the new ML-SSO with splicing strategy requires only 1.97 s. Finally, the highly efficient ML-SSO is employed for the inverse design of active shape changes of 4D-printed lattice structures. Our approach thus offers an intelligent design-fabrication paradigm for 4D printing of various shape-morphing AC structures.

CRediT authorship contribution statement

Xiaohao Sun: Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Luxia Yu:** Visualization, Investigation, Writing – review & editing, Methodology. **Liang Yue:** Methodology. **Kun Zhou:** Writing – review & editing, Methodology. **Frédéric Demoly:** Writing – review & editing, Methodology. **Ruike Renee Zhao:** Writing – review & editing, Methodology. **H. Jerry Qi:** Writing – review & editing, Supervision, Resources, Funding acquisition, Conceptualization, Methodology, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

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.

Appendix A. More information on the RNN-ML model

A.1. Construction of the RNN model

Our network architecture consists of a sequence input layer, an LSTM (Hochreiter and Schmidhuber, 1997) 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 implementation, training, and testing are conducted using MATLAB (2020a, MathWorks, Natick, MA). Before the training, all the input and output data are normalized through $x' = (x\text{-}mean(x))/std\,(x)$, 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″) show 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 a normalization is found to improve the network performance. The hidden size (number of neurons of each neural layer) of LSTM is set as 500, and the LSTM layer has $200 \times (500+N_y+1)$ learnable parameters. LSTM can process a sequence input of any lengths, and different time steps of LSTM share the same weights ("temporarily" invariant).

A.2. Hyperparameters for the training and network architectures

We perform a systematic study on the hyperparameters for the network architecture and training. The adopted hyperparameters for the training are listed here. The initial learning rate is set to 0.005, which decreases by multiplying a factor of 1/2 every 50 epochs. The training stops after the validation loss converges. The mini-batch size during training is set to 64. The adaptive moment estimation (Adam)[30] optimizer is used to train the network.

The studied hyperparameters for training include the hidden size (Fig. A.1a), initial learning rate (Fig. A.1b) and mini-batch size (Fig. A.1c). These hyperparameters do not significantly affect the model performance. We also study different RNN architectures, such as stacked (multi-layer) LSTM, bi-directional LSTM, and gated recurrent unit (GRU) network (an RNN with a different gating mechanism). These architectures again do not significantly affect the model performance (Fig. A.2).

In addition, we compare the performance of three ML models with different network architectures: RNN, CNN and NN. The CNN architecture and training parameters are described in our previous work (Sun et al., 2022), which is similar to that of the regression CNN used in Zhang and Gu (2020). The NN consists of five fully-connected layers with the hidden size of 100. The three models are trained and tested using the same datasets. Fig. A.3 shows the density scatter plots comparing the ground-truth and predicted values of x and y on the test set across the three ML models. The R^2 value of RNN is calculated using the values of both x and y, which is thus slightly higher than those presented in Fig. 1. The results demonstrate that RNN achieves much higher accuracy than CNN and NN.

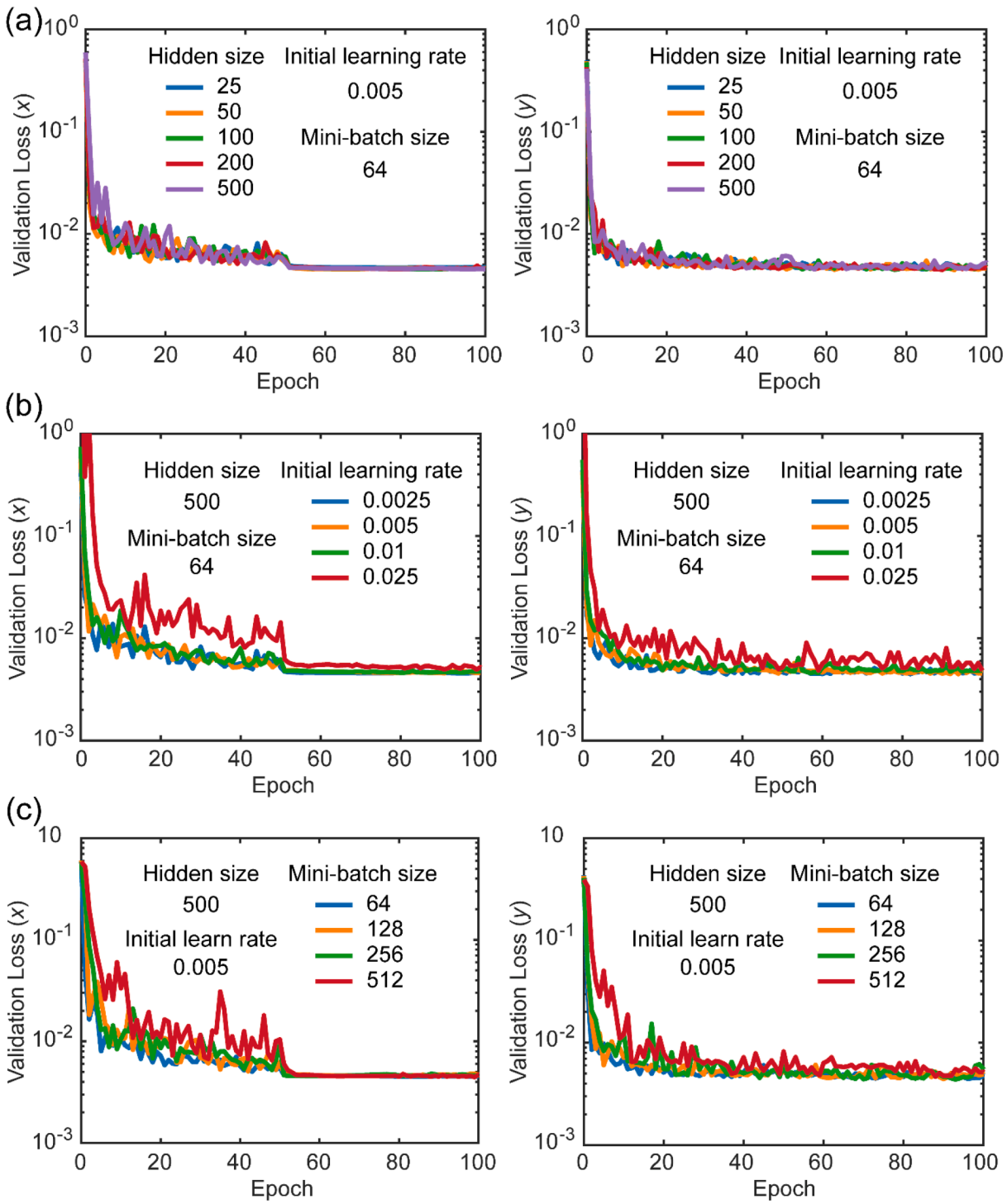

Fig. A.1. Validation loss (left for x and right for y) versus training epoch curves of the LSTM with different (a) hidden sizes, (b) initial learning rates, and (c) mini-batch sizes. No significant difference is found between different hyperparameters. The adopted ML model is based on the following hyperparameters: hidden size = 500, initial learning rate = 0.005, and mini-batch size = 64.

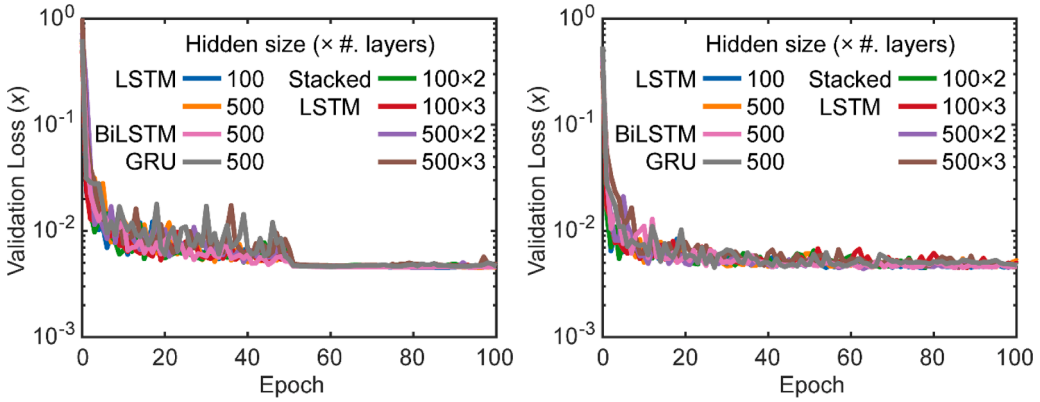

Fig. A.2. Validation loss (left for x and right for y) versus training epoch curves for different RNN architectures: LSTM, stacked LSTM, bi-directional LSTM, and GRU, with different hidden sizes and/or number of layers. No significant difference is found between different architectures. The adopted ML model is based on the LSTM architecture.

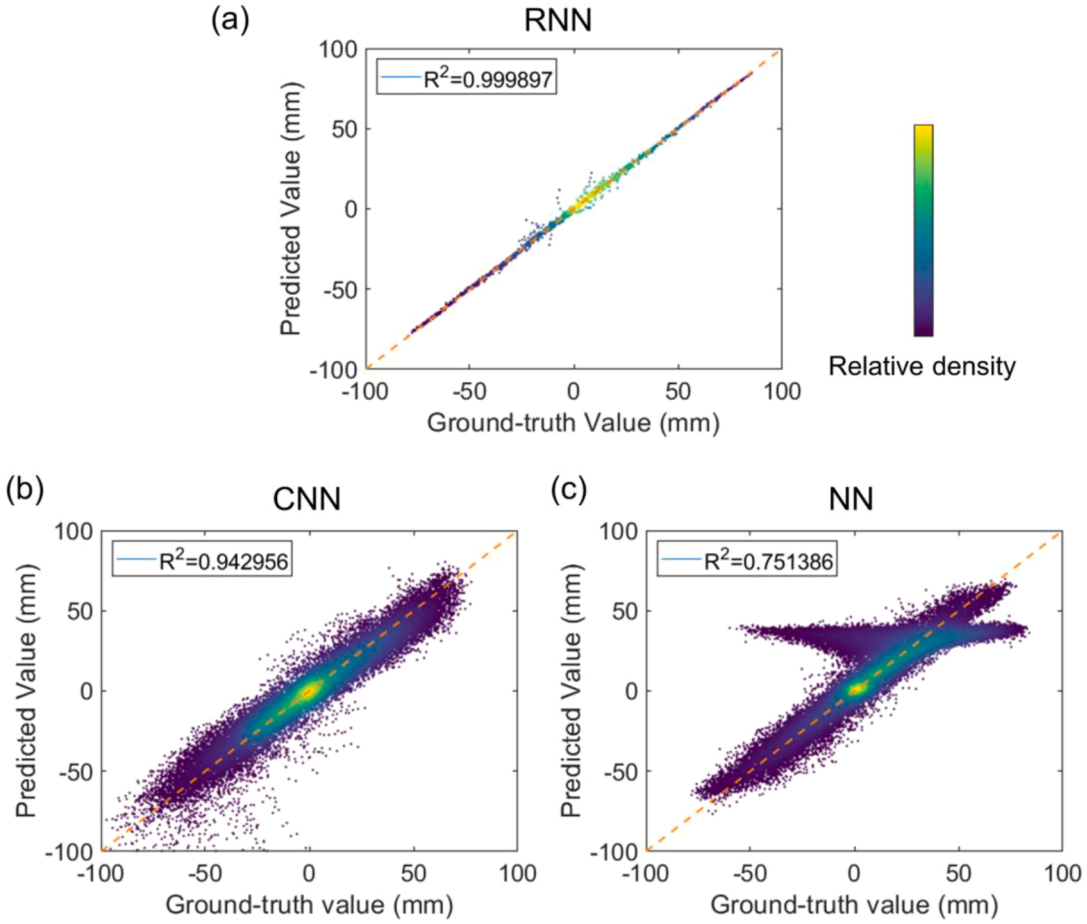

Fig. A.3. Performance of different network architectures: (a) RNN, (b) CNN, (c) NN. For each architecture, the density scatter plot comparing the ground-truth and ML-predicted coordinates (x and y shown together) on the test set is shown.

A.3. Error analysis of the ML model

We show the statistical distribution of the prediction errors for the test set (on which Fig. 1c is plotted) in Fig. A.4a, and the ground-truth and ML-predicted shapes of the 10 worst cases in Fig. A.4b. If we remove the 10 worst cases (only 0.36 % portion) from the entire test set (size is 2800) and re-plot the density scatter graph, nearly all points are concentrated on the regression line (Fig. A.4c). These results demonstrate that: (1) the datapoints with significant errors in Fig. 1c are only a tiny portion of the entire test set (<0.36 %); (2)

even for these cases, the largest errors typically happen at the end of the beam, whilst the accuracy is very high near the beam origin (black symbols, Fig. A.4b). The probability of significantly inaccurate predictions is thus very low in ML-SSO.

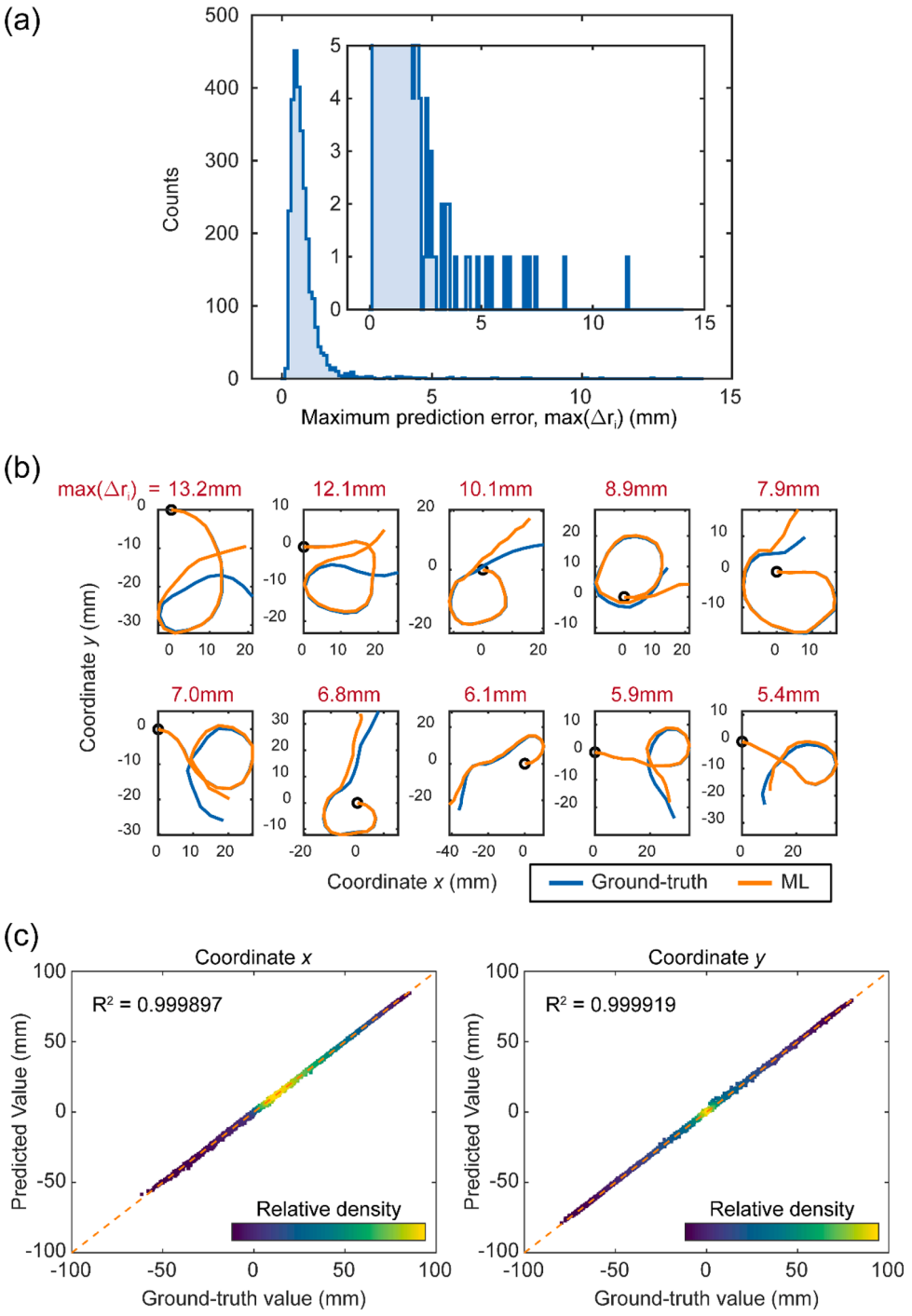

Fig. A.4. Study of prediction errors of the ML model. (a) Distribution of the maximum prediction error of the ML on the test dataset. The test set has 2800 datapoints, and each datapoint represents one entire beam, which consists of a material distribution and its actuated shape. For a particular datapoint, the prediction error Δr_i, where $i = 1, 2, \ldots, N_x$, is defined as the absolute distance between the i th sampling point of the ground-truth shape and that of the ML-predicted shape. (b) Ground-truth and ML-predicted shapes of the 10 worst cases with the highest $\max(\Delta r_i)$ values in the test set. Black symbols represent the origins (fixed end) of the beams. (c) Density scatter plots of the ground-truth versus ML-predicted values of the coordinate x (left) and y (right) for the test set with the 10 worst cases removed.

A.4. Optimized ML-prediction speed for ML-SSO

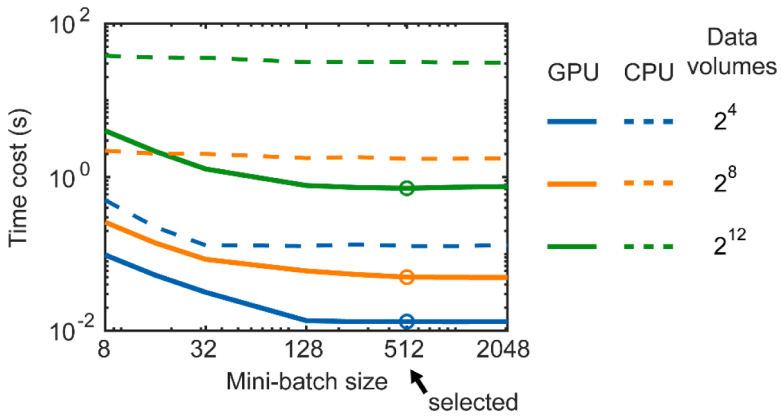

Fig. A.5. Time cost of ML predictions for different data volumes and mini-batch sizes using GPU (NVIDIA Quadro P620) or CPU (Intel Core i9-10900). This mini-batch size is used for the ML prediction instead of training. The optimal mini-batch size of 512 and the GPU device lead to fastest ML predictions and are thus adopted in our ML-SSO process.

Moreover, to fully exploit the parallel computing capability of ML and improve the optimization speed, the shape evaluation process in ML-SSO is implemented through a vectorized approach, which allows all potential material distributions to be processed in parallel by the ML model. In addition, in ML-SSO, the number of all potential designs for each subdomain is 2^($N_y \times N_{sub}$). Therefore, we examine the time cost for ML prediction of different amounts of data with different batch sizes and computing devices (Fig. A.5), and adopt the optimal batch size of 512 and the GPU device for all N_{sub} values in our ML-SSO optimizations. The achieved prediction speed is higher than that of our previous work (Sun et al., 2022). These strategies exploit the ML capability to deliver ultrafast and massive predictions.

A.5. Method of gradient calculation for Fig. 1d

The gradients of actuated coordinates **x** or **y** with respect to the voxel encoding can be written as

$$\partial \mathbf{x} / \partial \mathbf{M} \text{or} \partial \mathbf{y} / \partial \mathbf{M} \tag{5}$$

Since **x** and **y** are functions of **M** as described by which involves differentiable operations only, we use automatic differentiation (AD) to evaluate the gradients in Eq.(5) through the *dlgradient* function in MATLAB. As defined in Section 2.1, **M** is an array composed of "1″ and "0″. Here the component of **M** is denoted by M_v, which represents the material encoding of voxel v, where $v \in [1, N_y \times N_x]$ represents the voxel number. The gradient values, $\partial x_i/\partial M_v$ and $\partial y_i/\partial M_v$, for $i = 12$ and 24, based on an all-passive state (M_v=0 for any $v \in [1, N_y \times N_x]$), are then displayed in Fig. 1d.

Note that although the material variables (M_v) must be discrete values to be physically meaningful, they are treated as continuous variables when processed through the ML model. Since the ML model involves differentiable operations only, it naturally allows for the efficient computation of gradients via AD. Although the obtained gradients are with respect to non-continuous variables, they can provide high-level physical insights on the influence of each voxel input on all voxels' coordinate output (e.g., how changes in certain voxels affect the deformed shape)[28].

A.6. Direct extrapolation of RNN on inputs with N_x>24

Here, we test the extrapolation ability of our RNN model on input sequences with $N_x > 24$, i.e., use RNN to make direct predictions on these sequences. The results are compared with those obtained by the ML-splicing approach. As shown in Fig. A.6, the RNN model (which is trained on a dataset with N_x=24) does have the accuracy degradation but would not immediately lose the accuracy as N_x exceeds 24. It can remain accurate for N_x up to 32. This implies that RNN has learned some universal relations between consecutive voxel columns.

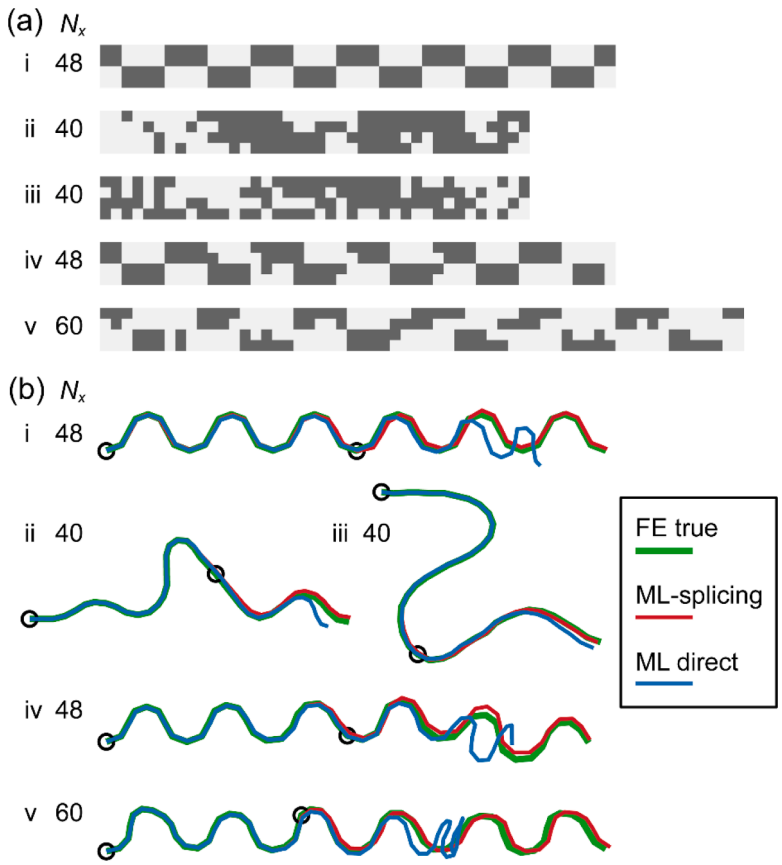

Fig. A.6. (a) Example material distributions with N_x>24, taken from (i) Fig. 6b and (ii - v) the optimized material distributions of cases in Fig. 7. (b) The corresponding FE ground-truth shapes, the ML-splicing shapes, and the ML-predicted ("ML-direct") shapes for patterns in (a). In each case, the two black symbols indicate the beam origin and the location at $N_x = 24$, respectively. The results show that the ML prediction (or extrapolation) can maintain the high accuracy for N_x up to 32, whilst ML-splicing is much more robust.

Appendix B. Experimental methods

B.1. Materials and 4D printing

The photocurable resin is prepared by mixing isobornyl acrylate (IOBA, Sigma-Aldrich) and aliphatic urethane diacrylate (AUD, Ebecryl 8402, Allnex, GA, USA) in a weight ratio of 1:1. Then, 1 wt% photoinitiator (Irgacure 819, Sigma-Aldrich), 0.08 wt% photo absorber (Sudan I, Sigma-Aldrich), and 0.04 wt% fluorescent dye (Solvent green 5, Orichem International Ltd., Hangzhou, Zhejiang, China) are added. The resin is thoroughly mixed before printing.

The grayscale digital light processing (g-DLP) printing technique (Yue et al., 2023) is used to print the designed structure, where the degree of curing (DoC) can be locally controlled by the assigned light intensity. Our ML-SSO designs are transformed into grayscale printing slices, where the active ("1″) and passive ("0″) phases correspond to grayscale percentages of 0 % (hence higher light intensity) and 60 % (hence lower light intensity), respectively, which later spatially assign the high-DoC and low-DoC phases in the printed structure, respectively. The printed structure is then placed in an 80 °C oven for 8 h to facilitate monomer volatilization. The low-DoC phase contains more residual monomers that can volatize and thus shows more volume shrinkage than the high-DoC phase at elevated temperatures. The shrinkage strain mismatch of the two phases induces the shape transformation.

B.2. Modification of optimal designs

Note that the material properties in experiments are different from those used in FE simulations. Experimental characterizations show that the printed two material phases show a modulus ratio of 0.06, while the ML-EA design assumes the identical modulus for two constituent phases. Additionally, the practical expansion mismatch is identified to be 0.05, which is also different from that used in ML-EA (i.e., 0.1). Such issues can be resolved by retraining the ML model based on the FE data with practical material properties (expansion mismatch and modulus difference) and re-running the ML-EA. Here, instead of retraining a new model, we adopt a design

conversion strategy (Sun et al., 2022) to approximately compensate effects of property difference of the two phases on the shape change, i.e., the optimal designs are converted using the analytical curvatures of multi-layer composite beams. The effectiveness of this strategy has been validated by FE simulations and experiments, as detailed in our previous work (Sun et al., 2022).

References

Athinarayanarao, D., et al., 2023. Computational design for 4D printing of topology optimized multi-material active composites. npj Comput. Mater. 9 (1), 1.

Chen, C.-T., Gu, G.X., 2019. Effect of constituent materials on composite performance: exploring design strategies via machine learning. Adv. Theory Simul. 2 (6), 1900056.

Cheng, J., et al., 2022. Centrifugal multimaterial 3D printing of multifunctional heterogeneous objects. Nat. Commun. 13 (1), 7931.

Demoly, F., et al., 2021. The status, barriers, challenges, and future in design for 4D printing. Mater. Des. 212, 110193.

Ge, Q., Qi, H.J., Dunn, M.L., 2013. Active materials by four-dimension printing. Appl. Phys. Lett. 103 (13), 131901.

Geiss, M.J., et al., 2019. Combined level-set-XFEM-density topology optimization of four-dimensional printed structures undergoing large deformation. J. Mech. Des. (5), 141.

Gu, G.X., et al., 2018. Bioinspired hierarchical composite design using machine learning: simulation, additive manufacturing, and experiment. Mater. Horiz. 5 (5), 939–945.

Guo, K., et al., 2021. Artificial intelligence and machine learning in design of mechanical materials. Mater. Horiz. 8 (4), 1153–1172.

Hamel, C.M., et al., 2019. Machine-learning based design of active composite structures for 4D printing. Smart Mater. Struct. 28 (6), 065005.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Kingma, D.P., Ba, J., 2014. arXiv preprint.

Kuang, X., et al., 2019. Grayscale digital light processing 3D printing for highly functionally graded materials. Sci. Adv. 5 (5), eaav5790.

Li, X., et al., *Physics-supervised deep learning–based optimization (PSDLO) with accuracy and efficiency*. 2023. 120(35): p. e2309062120.

Maute, K., et al., 2015. level set topology optimization of printed active composites. J. Mech. Des. 137 (11), 111402.

Rawat, S., Shen, M., 2018. arXiv preprint.

Roach, D.J., et al., 2021. Utilizing computer vision and artificial intelligence algorithms to predict and design the mechanical compression response of direct ink write 3D printed foam replacement structures. Addit. Manuf. 41, 101950.

Sigmund, O., 2011. On the usefulness of non-gradient approaches in topology optimization. Struct. Multidiscip. Optim. 43 (5), 589–596.

Sossou, G., et al., 2019. Design for 4D printing: modeling and computation of smart materials distributions. Mater. Des. 181, 108074.

Sun, X., et al., 2022. Machine learning-evolutionary algorithm enabled design for 4d-printed active composite structures. Adv. Funct. Mater., 2109805. **n/a**(n/a).

Sun, X., et al., 2024. Perspective: machine learning in design for 3D/4D printing. J. Appl. Mech. 91 (3), 030801.

Tanaka, M., et al., 2023. Turing pattern-based design and fabrication of inflatable shape-morphing structures. Sci. Adv. 9 (6), eade4381.

Wang, C., et al., 2021a. A comprehensive review of educational articles on structural and multidisciplinary optimization. Struct. Multidiscip. Optim. 64 (5), 2827–2880.

Wang, L., et al., 2021b. Evolutionary design of magnetic soft continuum robots. Proc. Natl. Acad. Sci. 118 (21), e2021922118.

Wilt, J.K., Yang, C., Gu, G.X., 2020. accelerating auxetic metamaterial design with deep learning. Adv. Eng. Mater. 22 (5), 1901266.

Wu, S., et al., 2020. Evolutionary algorithm-guided voxel-encoding printing of functional hard-magnetic soft active materials. Adv. Intell. Syst. 2 (8), 2000060.

Yang, Z., Yu, C.-H., Buehler, M.J., 2021. Deep learning model to predict complex stress and strain fields in hierarchical composites. Sci. Adv. 7 (15), eabd7416.

Yue, L., et al., 2023b. Cold-programmed shape-morphing structures based on grayscale digital light processing 4D printing. Nat. Commun. 14 (1), 5519.

Yue, L., et al., 2023a. Single-vat single-cure grayscale digital light processing 3D printing of materials with large property difference and high stretchability. Nat. Commun. 14 (1), 1251.

Zhang, Z., Gu, G.X., 2020. Finite-element-based deep-learning model for deformation behavior of digital materials. Adv. Theory Simul. 3 (7), 2000031.

Zhao, Z., Zhang, X.S., 2022. Topology optimization of hard-magnetic soft materials. J. Mech. Phys. Solids 158, 104628.

Zolfagharian, A., et al., 2020. Topology-optimized 4D printing of a soft actuator. Acta Mech. Solida Sin. 33 (3), 418–430.

Zolfagharian, A., et al., 2021. 4D printing soft robots guided by machine learning and finite element models. Sens. Actuat. A 328, 112774.