ADVANCED THEORY AND SIMULATIONS
www.advtheorysimul.com

Finite-Element-Based Deep-Learning Model for Deformation Behavior of Digital Materials

*Zhizhou Zhang and Grace X. Gu**

Smart composite materials fabricated through 4D-printing methods are attracting enormous research attention for their ability to respond (typically deform) under external stimuli. The design process for such smart materials requires iterations of finite-element simulations that are computationally expensive. Recently, researchers have tried replacing numerical simulations with machine learning (ML) models to predict the output at a much higher speed. However, there exist very few studies that explore the model algorithm's expressive capacity and analyze the physical interpretation based on the problem. This paper focuses on using ML to predict the nonlinear deformation behavior of digital materials. Various problem construction approaches and model performance are compared and discussed. It is shown that clustering the materials helps improve the generalization of training and models that treat material features as an array of numbers still face difficulties to provide accurate predictions. Inspired by modern computer vision technologies, convolutional kernels outperform other methods by recognizing the material distribution patterns. The performance is further enhanced after reconstructing the regression problem into classification. Moreover, high-level material design information can be extracted from the model through a sensitivity analysis. This framework may greatly improve the response prediction and design process for 4D-printed smart materials.

The advancement of additive manufacturing (also known as 3D-printing) allows the fabrication of structures with complex shapes and material distribution.[1] Consequently, 4D-printing emerged as a novel research field where smart materials that can respond to external stimuli are designed and 3D-printed.[2] Different from typical composites (fibers, laminates), the material distribution of 4D-printed products cannot be easily parameterized. Thus, the concept of digital materials was introduced to treat composites as assemblies of material pixels.[3] On the other hand, digital materials increase the design parameters by orders of magnitude which greatly complicates the design space of composites, especially when conducting massive amounts of experiments and compensating for their noise can be costly and time consuming.

Z. Zhang, Prof. G. X. Gu
Department of Mechanical Engineering
University of California
Berkeley, CA 94720-1740, USA
E-mail: ggu@berkeley.edu

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/adts.202000031

DOI: 10.1002/adts.202000031

To more efficiently explore the design space, the finite element method (FEM) has been widely applied to simulate the mechanical response of digital materials with a good match to the physical world.[4] However, one critical drawback of FEM is its huge computation cost. A typical FEM solver requires the inversion of the assembled stiffness matrix whose time complexity grows at least quadratically (can be cubically depending on the condition number) in the number of nodes in the mesh. Recently, researchers are applying machine learning (ML) to compress and accelerate various FEM simulation processes.[5] These surrogate models realize real-time fast prediction, analytical feature gradient, and high-level interpretation (shown later in this work) which typical optimization methods cannot acquire.[6] However, there exist very few studies that rationalize the model selection and problem construction based on the material features and the desired prediction. These steps can greatly affect the performance of ML. Moreover, this paper will demonstrate the sensitivity analysis through which one can obtain high-level knowledge. This approach is widely applicable to digital materials work that requires numerical predictions.

The focus of this work is to develop ML surrogate models to match the numerical simulation 4D-printed digital materials and specifically, materials that swell as their active pixels. Active materials include hydrogels, shape memory polymers, and silicon elastomers, which can deform drastically under external moisture or heat.[7] Such digital materials possess high nonlinearity due to the vast difference between its constituents' modulus and swelling response. The target geometry is chosen to be a curved beam that can achieve large deformation (**Figure** 1a), yet simple enough for human design intuition to help validate the high-level information extracted from ML models. We want to make a note here that the methods introduced by this paper are not restricted to the beam geometry and can be applied to any digital materials. More details about the problem setups are discussed in the Experimental Section. This work aims to utilize the power of ML to accelerate the deformation prediction of digital materials. The inputs will be the encoded material distribution (an array of 1s and −1s) as illustrated in Figure 1a and the outputs will be the coordinates of the eight highlighted nodes after deformation. The paper will compare the performance of different ML models and

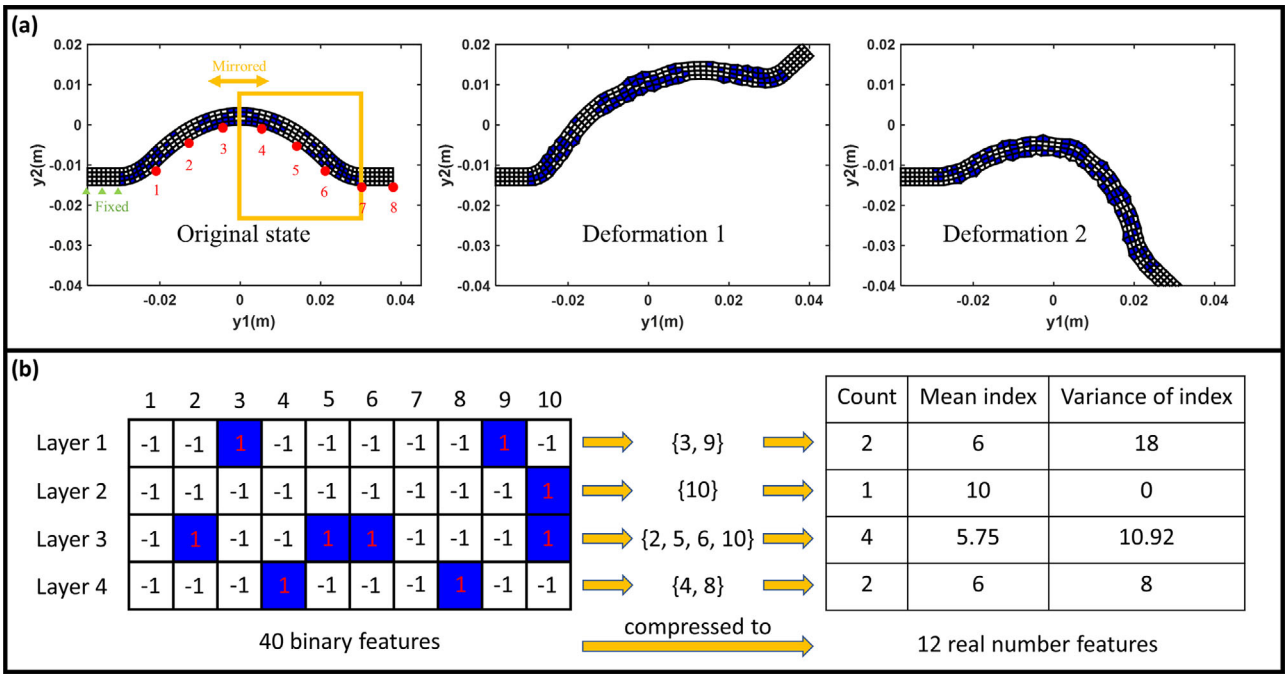

Figure 1. a) The initial state and the simulated deformation from two random material distributions. The geometry is fixed on its left tip and will deform once activated. Blue meshes represent the active materials and white meshes represent the passive materials. The 16 coordinates of the eight highlighted red nodes will be the prediction output of the ML algorithm. b) A sample feature compression process. Instead of storing the material of each element, the distribution is clustered. The active material elements on each of the mesh layers are clustered into one group. In each layer, the elements are indexed sequentially. Then, the count of active material elements, their mean index location, and variance of index location within each cluster is recorded as the compressed features.

analyze their compatibility with an FEM problem, and discuss approaches to extract high-level understanding from ML models. Moreover, we want to emphasize the importance of problem construction which may be more crucial than a good model. In the following parts of the paper, root mean square error (RMSE) is used to evaluate the ML models. For this deformation prediction problem, RMSE physically represents the distance between the predicted nodal coordinate from the ML model and the simulated ground truth coordinate. The loss function has two alternatives: the average loss $\frac{1}{8}\sum_{n=1}^{8}\sqrt{(y1_n-\widehat{y1}_n)^2+(y2_n-\widehat{y2}_n)^2}$ which measures the average RMSE over the eight highlighted nodes as in Figure 1a or the right tip loss $\sqrt{(y1_8-\widehat{y1}_8)^2+(y2_8-\widehat{y2}_8)^2}$ that only measures the RMSE at node 8 (node 8 is of special interest for its largest displacement variation over all the eight nodes). $y1_n$ and $y2_n$ are the predicted horizontal and vertical coordinates of the node n and $\widehat{y1}_n$ and $\widehat{y2}_n$ are the corresponding ground truth labels. All ML models are trained on the average loss except for the convolutional neural network (CNN) classification model.

The k-nearest neighbors algorithm (KNN) is first attempted as a baseline for the subsequent work. For KNN, the training process is just storing all the training data. A prediction is calculated as a weighted sum of the labels of the top k training data with the most similar input features to the test point. The weights for labels are calculated as the normalized inverse distance from each of the k training points to the test point. **Figure** 2a shows the average validation loss at different k values where $k = 5$ gives the best average validation loss of 7.7 mm based on the compressed 12 feature inputs and 8 mm based on the raw 100 feature inputs. The loss at node 8 is approximately twice the average loss over the 8 nodes which is around 1.6–1.7 cm. Considering the overall size of the geometry (less than 8 cm in $y1$ direction) and the range of node 8 displacement (less than 8 cm), the result from KNN is insignificant. This outcome is not surprising when a lazy algorithm like KNN is applied which normally requires a dense training set for good performance. Nevertheless, this validation loss of 7.7 mm still offers a preliminary starting line for the evaluation of more complex models.

Linear regression serves as the most widely applied data learning algorithm for its efficient closed-form training process yet moderate expressive power. Due to the nonlinearity of the displacement response of the digital material, quadratic $k(x_i, x_j) = (1 + a(x_1 \cdot x_2))^2$ and radial basis function (RBF) kernels $k(x_i, x_j) = \exp(-b\|xi - xj\|^2)$ are applied to increase the complexity of a linear model. Given the mean square error cost function, the closed-form prediction for a test point z from the ordinary least square (OLS) linear regression is given as $z(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$, and the closed-form prediction from kernelized linear regression is given as $[k(x_1, z) \ldots k(x_n, z)](\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\mathbf{Y}$ where x_is are the training data, $\boldsymbol{X}$ is a matrix with the training data stacked in rows, $\mathbf{Y}$ is a matrix with the training labels stacked in rows, $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{K}$ is the gram matrix whose (i, j)th entry is equal to $k(x_i, x_j)$. The weight penalty λ is set to 0.01, quadratic kernel constant a is set to be 1/900, and the RBF kernel constant b is set to 0.00065 for the compressed 12 feature inputs. These hyperparameters are tuned according to the average validation loss on the 1000 mini data batch.

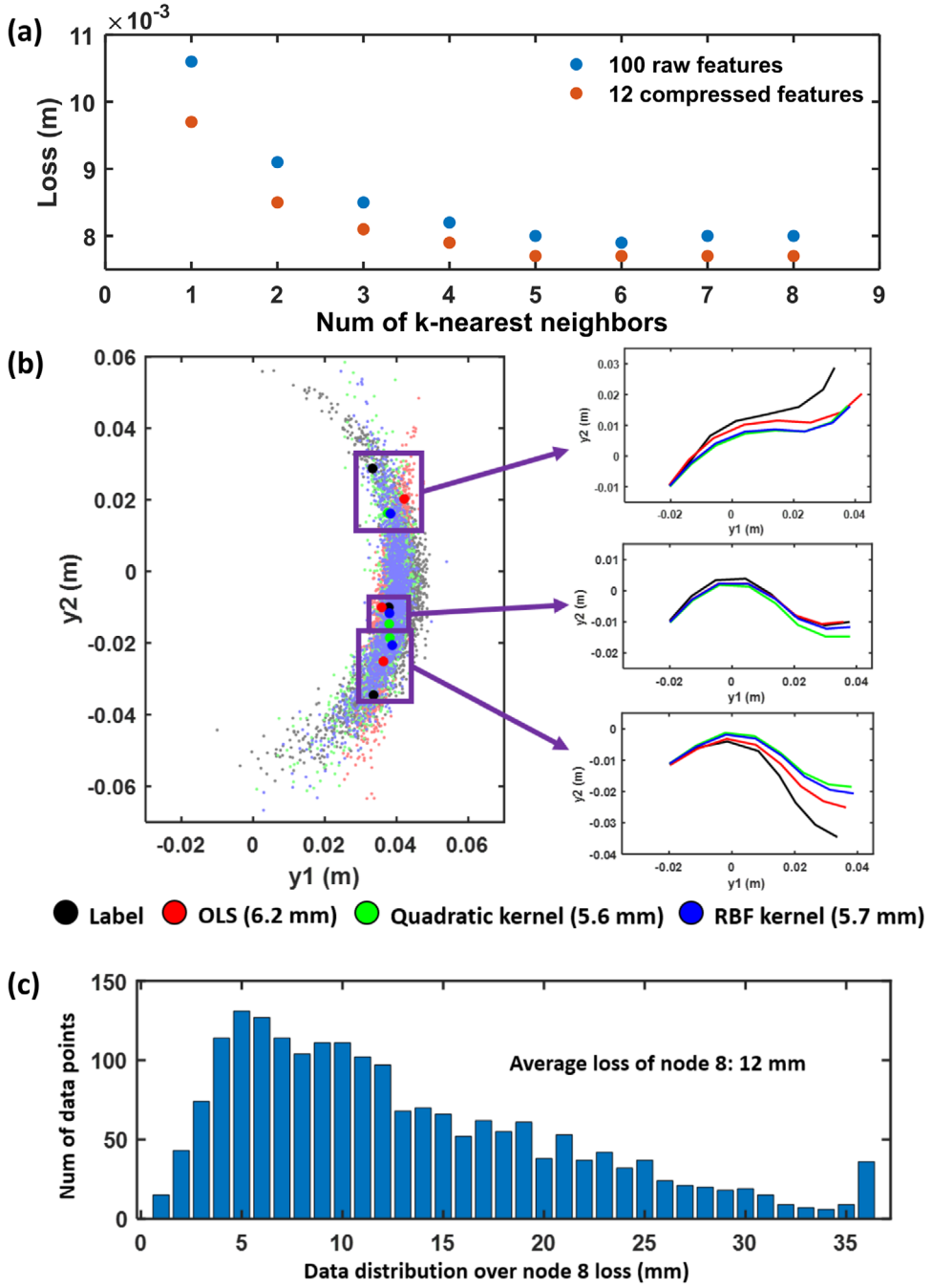

Figure 2. a) Average validation loss for KNN at different *k* values. b) Validation results of linear regression. The point cloud plot compares the ground truth (black) coordinate of node 8 with the validation results from ordinary least square (red), quadratic kernel (green), and Gaussian kernel (blue) linear regression. The average validation loss are 6.2, 5.6, and 5.7 mm for the three linear regression models. Three groups of predictions are selected randomly from the validation set with all the eight predicted nodal coordinates plotted on the right. c) The bar plot shows the validation data distribution from the quadratic kernel. Node 8 loss (12 mm) is more than twice of the average validation loss (5.6 mm).

The validation results of node 8 displacement from OLS, quadratic kernel, and RBF kernel linear regressions on the compressed features are shown in Figure 2b. To better visualize the predictions, three validation points are picked from the dataset with the predictions of all the eight nodes plotted on the right. The average validation loss over the 10 000 standard dataset of the three linear regression models (OLS, quadratic kernel, RBF kernel) are 6.2, 5.6, and 5.7 mm which are much improved upon the KNN results. The same training process is also tested on the 100 raw features which achieves the average validation loss of 6.3, 6.2, and 6.2 mm. Therefore, we can safely conclude that given a sparse dataset, feature compression through material clustering

does help improve the generalization of the ML algorithms through blurring the details of the features. However, the result from quadratic kernelized regression cannot be considered accurate since more than half of the validation data have node 8 distance loss of at least 10 mm as seen in Figure 2c. There are three potential reasons for the low accuracy from linear regression models. First, there are not enough data points to fully utilize the model's expressive power. Second, the complexity of the model needs to be enhanced. Or lastly, the problem is not well constructed. However, linear regression's average validation loss over the 1000 mini data batch is already below 6.1 mm which indicates only 8.2% decrease in loss for ten times the amount of data. Therefore, instead of collecting millions of data points, the paper will discuss in the following paragraphs how nonlinearity is added to raise model complexity and how the input features and output space are reconstructed to improve the model performance.

One major limitation of linear regression models is the linear dependence between the output responses and the features. In other words, each input feature has a fixed contribution to the output. To convert this constant relationship into a functional dependency, a neural network model is tested on this problem which adds in nonlinearity using activation functions. After fine-tuning on the 1000 mini data batch, the final architecture of the neural network contains an input layer with 12 compressed features, a hidden layer with 32 neurons, and an output layer with 16 coordinate predictions. The hidden layer is followed with a batch-norm layer and the rectified linear unit activation. The training is accomplished after 60 epochs of stochastic gradient descent with a batch size of 16. The initial learning rate is set to 0.001 which decreases to 0.0004 after 30 epochs. The average validation loss over the 10 000 standard dataset is 5.3 mm which slightly outperforms the linear models, but still not even close to the desired 1.4 mm average loss (detailed in the Experimental Section).

Despite the large validation loss, the training loss of the neural network has reached below 0.5 mm indicating a heavy overfitting. We want to make a note here that even after an overfitting is observed (training loss goes below validation loss), the validation loss keeps decreasing and converges at 60 epochs. This suggests the possibility of better reconstructing the problem. By far, the abovementioned ML models treat the material distribution as an array of numbers (features). In this case, the features are considered unordered where one would expect an exact same model performance if the order of the features is changed consistently for training and validation data. However, when solving such problems in an FEM approach, the stiffness matrix of an element is heavily coupled with its neighbor elements. Thus, instead of an unordered array of numbers that discards the spatial information, the material distribution should be treated as a tensorial object which has an underlying spatial basis. This is similar to the case of image recognition in the computer vision field where both the color intensity and the spatial arrangement of image pixels should be considered in the model.

CNN is the most widely used framework in computer vision for its ability to identify local image patterns with convolution kernels. **Figure** 3a demonstrates three special cases of material pattern which one would expect to bend downward, extend horizontally, and bend upward after the blue material elements are activated. And the three kernels are specialized so that they will produce the largest output 9 only if the corresponding target pattern shows up. Otherwise, the output from the kernel will be much smaller depending on how different the pattern is. In the real problem, material distribution is not intuitive as the samples given, nevertheless, the kernels can be learned through the training process of CNN. Figure 3b shows the fine-tuned CNN regression architecture which achieves an average validation loss of 5.5 mm over the mini dataset and 2.4 mm over the standard dataset. Different from previous models, CNN shows a huge leap in its performance after the dataset expansion. To further explore the model's capability, CNN is then tested on the large dataset with 40 000 training points and 10 000 validation points. The training is accomplished after 20 epochs of stochastic gradient descent with a batch size of 16. The initial learning rate is set to 0.001 which decreases to 0.0002 after 10 epochs. The CNN regression model achieves an average validation loss of 1.49 mm (R-squared value of 97%) over the large dataset which almost meets the goal of 1.4 mm. To avoid any data contamination during gradient descent, the model is further evaluated on the test dataset which gives an average loss of 1.47 mm.

CNN regression architecture has proved its high capability for such elementwise input features. However, the regression approach often suffers from its poor generalization, especially for complex models. As seen in Figure 3b, the regression architecture is kept relatively simple to avoid overfitting. Therefore, the problem is reconstructed into multiclass classification with a crossentropy loss which measures the difference in probability distribution between a prediction and a ground truth label. The reconstruction process is illustrated in Figure 3d. The real number axis is discretized into small regions which are labeled as different classes during the training process. The size of the small regions is chosen to be 1 mm for this problem. Through the reconstruction, instead of directly predicting the nodal coordinate after activation, the model predicts the probability that a node would stay in a region. The class labels are mapped back to the real numbers at the center of each region during the prediction. Different from typical classification problems whose class magnitude is physically meaningless, the order and distance between classes contain considerable information in this reconstructed problem. Thus, the real number coordinate prediction is calculated as the weighted sum over the classes. The training process is the same as for the CNN regression model.

Since the 16 coordinates of the eight nodes each has a different range, 16 different CNN architectures are required for all the outputs which can be cumbersome. Therefore, the reconstruction approach is first tested on the $y2$ coordinate of node 8 which possesses the largest variance among all the 16 coordinates. From a preliminary observation on the training dataset label range, the real $y2$ axis is divided into 125 regions: one region for $(-\infty, -4.7\text{ cm})$, 123 regions for $[-4.7\text{ cm}, 7.6\text{ cm}]$, and one region for $(7.6\text{ cm}, +\infty)$. Figure 3c shows the fine-tuned CNN classification architecture which achieves a validation loss of 1.89 mm ($|y2_8 - \widehat{y2_8}|$) with an R-squared value of 98%. Note that the CNN regression model's node 8 $y2$ validation loss is 2.74 mm. The reconstruction approach is then tested on the $y1$ coordinate of node 8 to show its general applicability. The new architecture for node 8 $y1$ reaches a validation loss of 1.11 mm ($|y1_8 - \widehat{y1_8}|$). To better visualize the performance, the distance

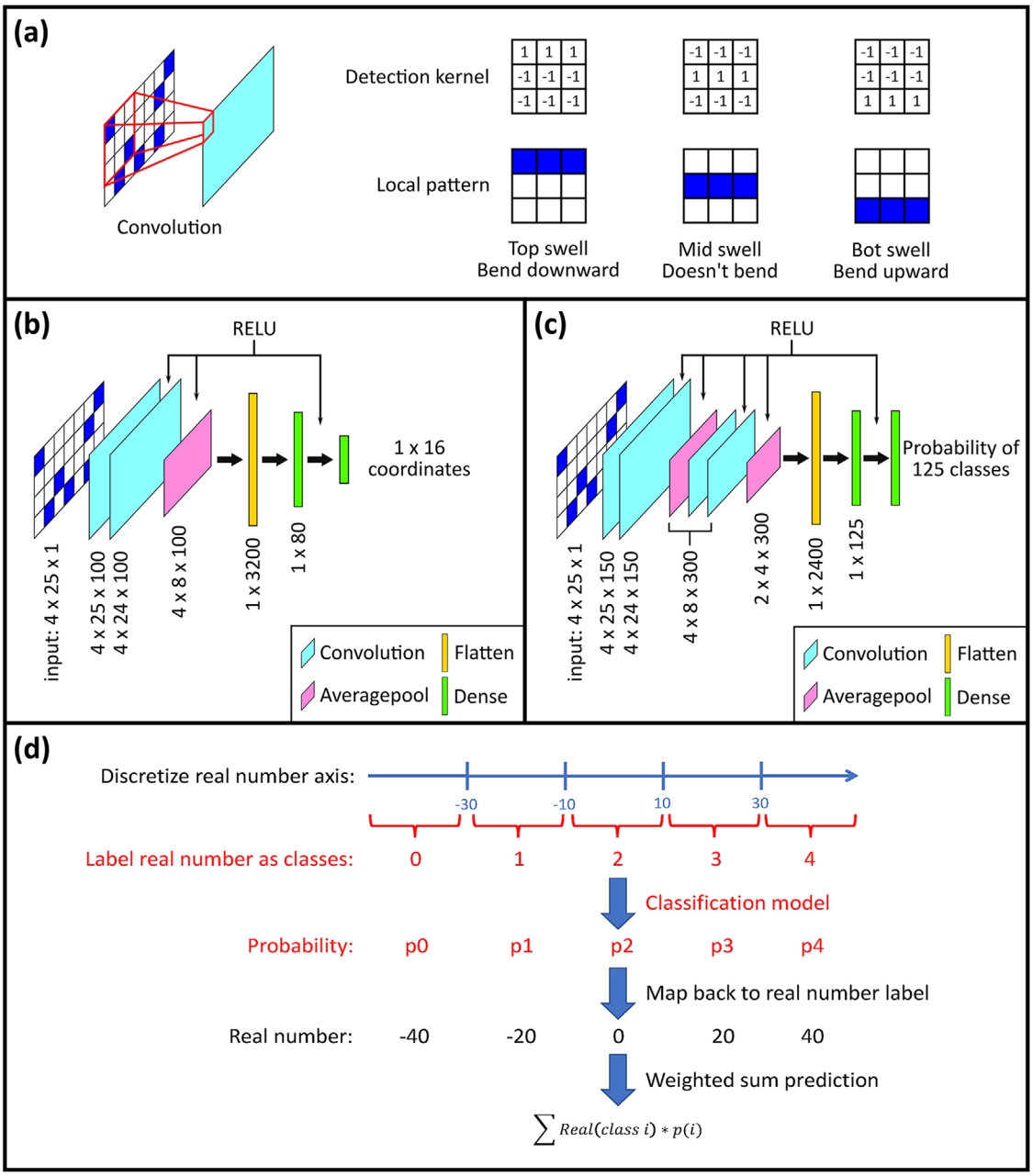

Figure 3. a) Convolution kernels learn to recognize material distribution patterns. Some sample detection kernels are shown on the right. Each sample kernel will produce an output of 9 if a corresponding target pattern shows up. Otherwise, they will produce much smaller outputs. b) The architecture of the fine-tuned CNN regression model that predicts the 16 coordinates. c) The architecture of the fine-tuned CNN classification model that predicts node 8 *y2* coordinate. d) The regression problem can be reconstructed into a classification problem. During the labeling process, the real number coordinates should be discretized into small regions and each of them represents a class label. The model is then trained to predict the probability of each class (highlighted in red). Thereafter, the probability outputs can be mapped back to a real number by taking a weighted sum.

loss of node 8 from CNN regression and classification models are plotted in **Figure** 4a,b. Both plots show great advancement compared to the quadratic kernelized linear regression model, and the predictions from the CNN classification model concentrate more on the low error region. The reconstructed models reach a node 8 *y1* and *y2* loss of 1.12 and 1.89 mm on the test dataset indicating no overfitting during training or validation. Despite the undesirable computation cost, the success of the classification model proves the importance of problem construction besides model selection when applying ML techniques to finite element problems.

One major drawback of a deep neural network (compared to linear regressions) is its lack of interpretability due to the complex architecture. Although it is very difficult to fully understand how the model predicts, a sensitivity analysis can still provide some high-level ideas for the input–output relationship. In this work,

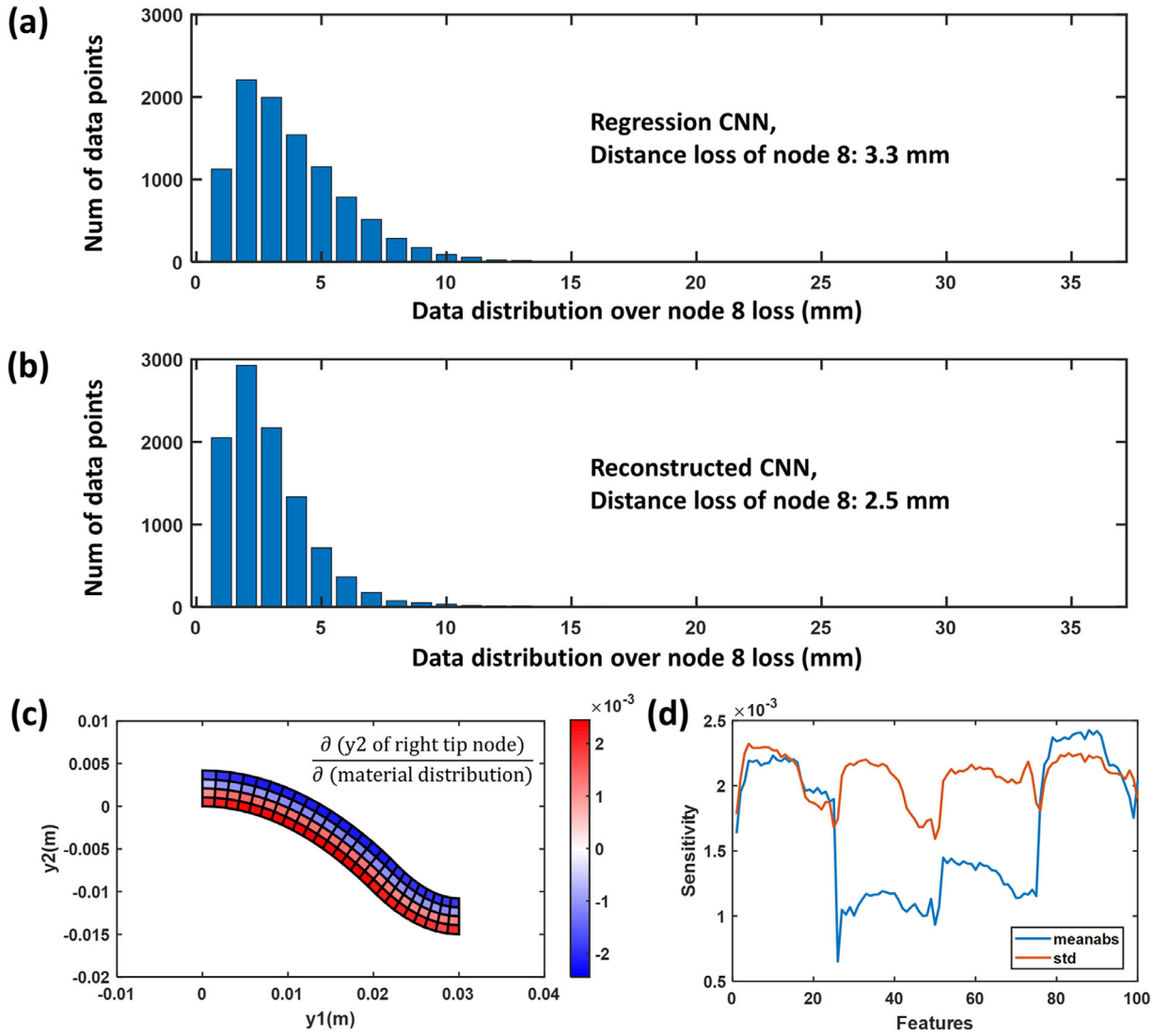

Figure 4. a) The bar plot shows the validation data distribution from the CNN regression model. It has an average validation loss of 1.49 mm and node 8 loss of 3.3 mm. b) The bar plot shows the validation data distribution from the reconstructed CNN classification model with node 8 loss of 2.5 mm. c) The expected sensitivity of node 8 *y2* coordinate with respect to the material elements. The expected sensitivity is taken to be the average sensitivity over the 10 000 validation points. For elements in red, they are expected to increase *y2* (right tip moves up) if they are assigned to be active materials. And for elements in blue, they are expected to push down the right tip if they are assigned to be active. d) The mean absolute value and the standard deviation of the expected sensitivity over the 10 000 validation points.

sensitivity is defined as the gradient of an output direction over an input feature[8] without square or absolute value. In the case of a linear model, the sensitivity is simply the constant weight in front of a certain feature (either original or augmented). While for neural networks, a feature's "weight" becomes a function that depends heavily on the entire feature set and can be computed through the backpropagation. As a result, calculating the sensitivity based on a specific data point is not generally informative for the problem. Instead, the average sensitivity over 10 000 validation data is calculated and plotted in Figure 4c to interpret the reconstructed CNN model that predicts node 8 *y2* displacement. Since the active material elements are encoded as 1 and −1 otherwise (detailed in the Experimental Section), for elements in red, they are expected to push *y2* upward if they are assigned the active material. Similarly, for elements in blue, they are expected to push *y2* downward if they are assigned to be active. The simplicity of the geometry allows validating the sensitivity plot with physical intuition. One would expect the same effect from the active materials as plotted in Figure 4c. Note that the sensitivity does not change much along the *y1* direction since the material distribution is mirrored about the center axis. This consistency between sensitivity analysis and intuition proves the feasibility of this approach to gain high-level physical understanding. This would manifest more in other complex geometries which will not be discussed in this work. Figure 4d plots the standard deviation of sensitivities and the mean of absolute sensitivities among the 100 material features. The standard deviations have similar or larger amplitudes than the mean values implying high

Table 1. The validation results of different ML models with and without feature compression, all units are in millimeter. Note that the two CNN models are trained on the 50 000 large dataset for their outstanding model capacity, while all other models are trained on the 10 000 standard dataset.

	KNN	OLS	Quadratic	RBF	Neural network	CNN regression	CNN classification
Average loss (100/12 features)	8.0/7.7	6.3/6.2	6.2/5.6	6.2/5.7	6.0/5.3	1.5/(N/A)	N/A
Node 8 loss (100/12 features)	18.1/17.2	13.9/14.2	13.5/12.1	13.8/12.4	13.1/11.7	3.3/(N/A)	2.5(N/A)

nonlinearity of this displacement prediction problem as mentioned previously.

To summarize, this work demonstrates the capability of machine learning models in predicting and understanding the deformation response of digital materials. The performances of ML models are summarized in **Table 1**. Results show that models that treat element information as unordered arrays fail to accurately predict the deformation. Instead, significantly higher performance is achieved using convolution kernels which learn the material distribution patterns without a loss of spatial information. Furthermore, discretizing the output space into a classification model (crossentropy loss) improves the generalization and outperforms the regression model (mean square loss). However, the reconstruction approach comes with the highest computational cost for training (30 min for one model and a total of 16 models are required for all 16 coordinates) depending on the range of deformation and resolution of prediction. And all models take less than 0.01 s to make a prediction. The sensitivity analysis through backpropagation has proved its ability to offer high-level design information and measure the nonlinearity of the response function. Though this work focused on the deformation of a curved beam, it is believed that the ML framework can be applied to digital materials with more complex geometries and responses.

Experimental Section

Simulation Setup: The original state (nonactivated) of the curved beam geometry is shown in Figure 1a with its left end fixed. The geometry was meshed into 4*64 2D plane stress elements. Each element belonged either to the passive material (Modulus: 1 GPa, Poisson's ratio: 0.3, Swelling: 0%) or the active material (Modulus: 10 MPa, Poisson's ratio: 0.48, Swelling: 18%) with isotropic linear elasticity assumption under the finite deformation scheme. To constrain the complexity of the test problem without loss of generality, the following assumptions were taken: The design space was taken to be the material distribution within the highlighted zone (yellow) which contained 4*25 elements. The active material elements were selected randomly from a uniform distribution among the 100 candidate grids. The total number of active elements was upper bounded by 50 and lower bounded by 30. This material distribution was then mirrored about the symmetrical axis of the geometry. The simulation input files were generated with Matlab, solved with Abaqus, and postprocessed with the Abaqus2Matlab toolbox. Each simulation took ≈15 s on the Intel Xeon E-2146G central processing unit (CPU). In Figure 1a on the right of the original state plot, the plots (deformation 1 and deformation 2) show the deformation responses from two randomly generated sample material distributions.

Machine Learning: The problem setup described above yielded approximately 1e30 digital material configurations which were impossible to be enumerated. Therefore, machine learning will be applied to learn and predict the geometry deformation after the digital material is activated. Precisely, the input features will be the material distribution and the outputs will be the 16 coordinate displacements of the eight interest nodes as seen in Figure 1a. Three separate datasets (mini: 1000, standard: 10 000, large: 50 000) were generated for hyperparameter tuning, model comparison, and investigating the effect of dataset expansion. Each dataset had 20% of its data points isolated for validation. Different ML models will be tested and discussed on this problem including *K*-nearest neighbors, linear regression (kernelized), neural networks, convolutional neural networks (regression and classification). The goal was to reach an average validation loss of less than 1.4 mm which was approximately the diagonal length of an element.

Feature Compression: Since the 100 features of the original problem were all binary values, reducing them into fewer number of real value features might improve the generalization of the ML model. However, the uniform data generation process created a spherical data distribution in the input space where all features were expected to share the same variance. Therefore, instead of extracting the eigenvectors of the data matrix, it was chosen to compress the features through clustering which still retained the material distribution information, but on a larger scale. Like all other feature compression methods, clustering the element information will cause an increase in bias, however, less focus on a single element will help the generalization. For the 2D geometry with a curved beam shape that is studied in this work, one would expect a higher importance in the material distribution along the thickness direction than the length. A sample feature compression is illustrated in Figure 1b where a 4 by 10 element system is used instead of the real 4 by 25 system for concise visualization. A total number of four clusters were created with each of them representing the material distribution of one layer of element. For each layer, the elements were indexed from 1 to 25. Then, the count of active materials, the mean index, and the variance of index were recorded to represent the distribution yielding the 12 compressed features.

Acknowledgements

The authors acknowledge support from the Extreme Science and Engineering Discovery Environment (XSEDE) at the Pittsburgh Supercomputing Center (PSC) by National Science Foundation Grant Number ACI-1548562. Additionally, the authors acknowledge support from the Chau Hoi Shuen Foundation Women in Science Program and an NVIDIA GPU Seed Grant.

Conflict of Interest

The authors declare no conflict of interest.

Keywords

convolutional neural networks, deep learning, digital materials, finite elements, machine learning

Received: February 14, 2020
Revised: April 14, 2020
Published online: May 17, 2020

[1] a) M. Vaezi, S. Chianrabutra, B. Mellor, S. Yang, *Virtual Phys. Prototyping* **2013**, *8*, 19; b) G. X. Gu, C.-T. Chen, D. J. Richmond, M. J. Buehler, *Mater. Horiz.* **2018**, *5*, 939; c) Z. Jin, Z. Zhang, G. X. Gu, *Manuf. Lett.*

2019, *22*, 11; d) Z. Jin, Z. Zhang, G. X. Gu, *Adv. Intell. Syst.* **2020**, *2*, 1900130; e) W. Gao, Y. Zhang, D. Ramanujan, K. Ramani, Y. Chen, C. B. Williams, C. C. Wang, Y. C. Shin, S. Zhang, P. D. Zavattieri, *Comput.-Aided Des.* **2015**, *69*, 65.

[2] a) Z. X. Khoo, J. E. M. Teoh, Y. Liu, C. K. Chua, S. Yang, J. An, K. F. Leong, W. Y. Yeong, *Virtual Phys. Prototyping* **2015**, *10*, 103; b) Z. Zhang, K. G. Demir, G. X. Gu, *Int. J. Smart Nano Mater.* **2019**, *10*, 205.

[3] a) F. Momeni, X. Liu, J. Ni, *Mater. Des.* **2017**, *122*, 42; b) Y. Mao, K. Yu, M. S. Isakov, J. Wu, M. L. Dunn, H. J. Qi, *Sci. Rep.* **2015**, *5*, 13616; c) J. Hiller, H. Lipson, *Rapid Prototyping J.* **2010**, *16*, 241.

[4] a) Y. Yu, H. Liu, K. Qian, H. Yang, M. McGehee, J. Gu, D. Luo, L. Yao, Y. J. Zhang, *arXiv preprint arXiv:1909.02083* **2019**; b) Z. Zhao, H. J. Qi, D. Fang, *Soft Matter* **2019**, *15*, 1005.

[5] a) F. Martínez-Martínez, M. J. Rupérez-Moreno, M. Martínez-Sober, J. Solves-Llorens, D. Lorente, A. Serrano-López, S. Martínez-Sanchis, C. Monserrat, J. D. Martín-Guerrero, *Comput. Biol. Med.* **2017**, *90*, 116; b) C.-T. Chen, G. X. Gu, *MRS Commun.* **2019**, *9*, 556; c) M. Bessa, R. Bostanabad, Z. Liu, A. Hu, D. W. Apley, C. Brinson, W. Chen, W. K. Liu, *Comput. Methods Appl. Mech. Eng.* **2017**, *320*, 633; d) C. Yang, Y. Kim, S. Ryu, G. X. Gu, *Mater. Des.* **2020**, *189*, 108509; e) C. T. Chen, G. X. Gu, *Adv. Sci.* **2020**, *7*, 1902607; f) R. Hambli, H. Katerchi, C.-L. Benhamou, *Biomech. Model. Mechanobiol.* **2011**, *10*, 133; g) W. Yan, S. Lin, O. L. Kafka, Y. Lian, C. Yu, Z. Liu, J. Yan, S. Wolff, H. Wu, E. Ndip-Agbor, *Comput. Mech.* **2018**, *61*, 521; h) K. Wang, W. Sun, *Comput. Methods Appl. Mech. Eng.* **2018**, *334*, 337; i) F. Ghavamian, A. Simone, *Comput. Methods Appl. Mech. Eng.* **2019**, *357*, 112594; j) Z. Liu, C. Wu, M. Koishi, *Comput. Methods Appl. Mech. Eng.* **2019**, *345*, 1138; k) X. Lu, D. G. Giovanis, J. Yvonnet, V. Papadopoulos, F. Detrez, J. Bai, *Comput. Mech.* **2019**, *64*, 307.

[6] 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) J. H. S. Almeida,Jr., M. L. Ribeiro, V. Tita, S. C. Amico, *Compos. Struct.* **2017**, *178*, 20; c) H. A. Deveci, L. Aydin, H. Seçil Artem, *J. Reinf. Plast. Compos.* **2016**, *35*, 1233; d) C. T. Chen, G. X. Gu, *Adv. Theory Simul.* **2019**, *2*, 1900056.

[7] a) S. Liu, X. Chen, Y. Zhang, in *3D and 4D Printing of Polymer Nanocomposite Materials*, Elsevier, Doha, Qatar **2020**, p. 427; b) A. B. Baker, S. R. Bates, T. M. Llewellyn-Jones, L. P. Valori, M. P. Dicker, R. S. Trask, *Mater. Des.* **2019**, *163*, 107544; c) H. Wu, P. Chen, C. Yan, C. Cai, Y. Shi, *Mater. Des.* **2019**, *171*, 107704; d) M. López-Valdeolivas, D. Liu, D. J. Broer, C. Sánchez-Somolinos, *Macromol. Rapid Commun.* **2018**, *39*, 1700710; e) A. Miriyev, K. Stack, H. Lipson, *Nat. Commun.* **2017**, *8*, 596.

[8] G. Montavon, W. Samek, K.-R. Müller, *Digital Signal Process.* **2018**, *73*, 1.