ARTIFICIAL INTELLIGENCE

Mechanical neural networks: Architected materials that learn behaviors

Ryan H. Lee[1]†, Erwin A. B. Mulder[2], Jonathan B. Hopkins[1]*†

Aside from some living tissues, few materials can autonomously learn to exhibit desired behaviors as a consequence of prolonged exposure to unanticipated ambient loading scenarios. Still fewer materials can continue to exhibit previously learned behaviors in the midst of changing conditions (e.g., rising levels of internal damage, varying fixturing scenarios, and fluctuating external loads) while also acquiring new behaviors best suited for the situation at hand. Here, we describe a class of architected materials, called mechanical neural networks (MNNs), that achieve such learning capabilities by tuning the stiffness of their constituent beams similar to how artificial neural networks (ANNs) tune their weights. An example lattice was fabricated to demonstrate its ability to learn multiple mechanical behaviors simultaneously, and a study was conducted to determine the effect of lattice size, packing configuration, algorithm type, behavior number, and linear-versus-nonlinear stiffness tunability on MNN learning as proposed. Thus, this work lays the foundation for artificial-intelligent (AI) materials that can learn behaviors and properties.

INTRODUCTION

Scientists have been inspired by the interconnected network of neurons that constitute biological brains and enable complex learning with unmatched speed and energy efficiency. Consequently, many have sought to leverage a variety of interconnected networks to mimic natural learning for numerous artificial-intelligent (AI) applications (*1*–*3*).

Some of the first networks developed for AI purposes were purely mathematical in form. The concepts underlying these mathematical networks, called artificial neural networks (ANNs) (Fig. 1A), were first introduced by McCulloch and Pitts (*4*) but were later matured by Rosenblatt (*5*). The mathematical formulation underlying ANNs can be diagrammed using interconnected lines, shown in blue in Fig. 1A, that represent scalar values, called weights (*6*), which are multiplied by input numbers that are fed into multiple layers of activation functions (*6*), called neurons, which ultimately produce output values. If the ANN is provided with a set of known input and output values, then the network can be trained by tuning its weights so that it accurately predicts previously unknown output values that result for any desired input values. Hornik *et al.* (*7*) proved the true AI potential of ANNs by demonstrating that, with sufficiently large numbers of neurons and layers, ANNs could learn to model almost anything by accurately mapping any number of inputs to any number of outputs. Tuning the weights of sizeable ANNs, however, proved to consume large amounts of computational time and energy using traditional digital computers.

Thus, further inspired by the physical nature of biological brains, scientists began developing physical networks to more rapidly tune weights (i.e., learn) with higher efficiencies because of their analog nature. Most of these physical networks can be classified as electrical (*8*–*12*) or optical (*13*–*17*) networks. Although some physical neural networks use the vibrations of mechanical structures to improve the speed and efficiency of learning, none yet exists that is purely mechanical. Roboticists have learned to leverage the dynamics of mechanical bodies as a computational resource for enabling mathematical ANNs to be more efficiently trained by restricting only the final layer′s weights to be tuned. This approach, called morphological computation (*18*), is a mechanical version of the concept of reservoir computing (*19*, *20*), where the reservoir used to simplify the mathematical computation is the structure of the robot itself. Networks of springs and point masses (*21*, *22*), tensioned cables and rigid bodies (*23*, *24*), and soft bodies (*25*, *26*) have been used to demonstrate this approach. The most mechanical instantiation of a neural network to date was proposed by Hermans *et al.* (*27*). This network consists of a vibrating plate that is excited by acoustic waves as inputs and outputs. Instead of tuning the mechanical properties of the plate itself (i.e., its stiffness, damping, or mass properties) to tune the network′s weights, masking signals of interfering acoustic waves were electrically generated to train the network. This concept was recently extended by Wright *et al.* (*28*) using multiple layers of vibrating plates to achieve a deep physical neural network.

In this work, a different physical network, called a mechanical neural network (MNN), is introduced. MNNs are lattices of interconnected tunable beams, shown in blue in Fig. 1B, that join at nodes, which are driven by force or displacement inputs and outputs. The stiffness values of the interconnected beams are tuned as network weights to train the lattice such that it can learn desired mechanical behaviors (e.g., shape morphing, acoustic wave propagation, and mechanical computation) and bulk properties (e.g., Poisson′s ratio, shear and Young′s modulus, and density). Thus, this work introduces a class of architected materials (a.k.a., mechanical metamaterials) (*29*) that learn as a consequence of prolonged exposure to unanticipated ambient loading conditions. Although others have proposed acoustic metamaterials that can perform specific mechanical computations (*30*, *31*), these materials are not neural networks and thus cannot learn. Hughes *et al.* (*32*) proposed an acoustic metamaterial that behaves as a trained neural network, but a fabricated version of the proposed design could not

[1]Mechanical and Aerospace Engineering, University of California, Los Angeles, Los Angeles, CA 90095, USA. [2]Mechanics of Solids, Surfaces, and Systems, University of Twente, Enschede, Netherlands.
*Corresponding author. Email: hopkins@seas.ucla.edu
†These authors contributed equally to this work.

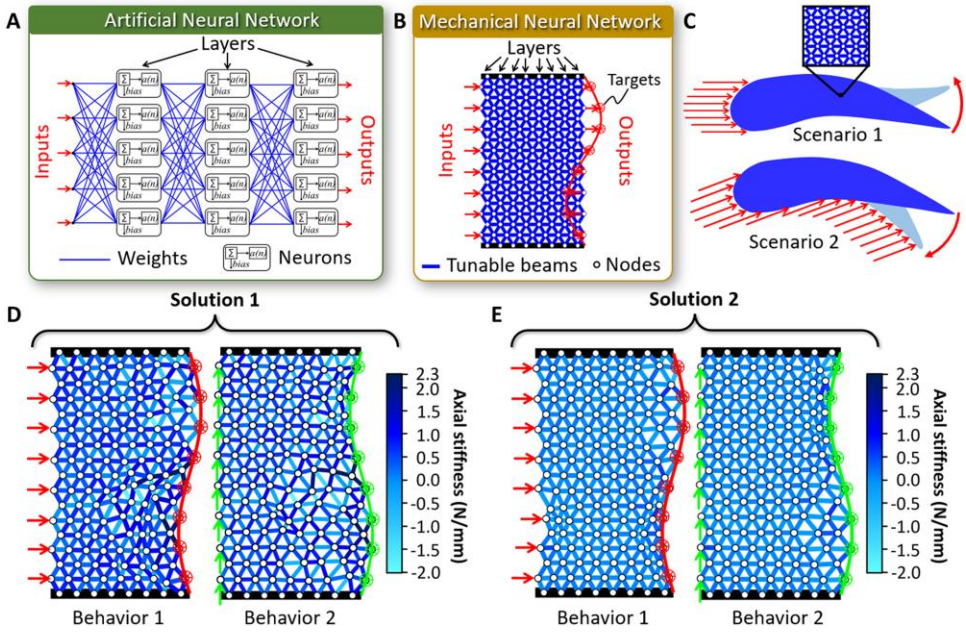

Fig. 1. Introduction to MNNs and how they learn mechanical behaviors. (**A**) ANNs mathematically map numerical inputs to outputs by tuning scalar weights within layers of neurons consisting of activation functions. (**B**) MNNs are mechanical analogs to ANNs in that they map force and displacement inputs and outputs using tunable beams, which are analogous to weights, and physical nodes, which are analogous to neurons. (**C**) Example shape-morphing behaviors that could be learned by MNN aircraft wings in two different scenarios. (**D** and **E**) Two different combinations of beam stiffness values (i.e., solutions) that achieve the same two shape-morphing behaviors.

learn new behaviors because training is performed during the design process by adjusting the mass within a vibrating plate using simulation. Although other mechanical concepts have also been proposed and demonstrated using simulation only (*33*, *34*), the MNN concept introduced here is physically demonstrated experimentally. The concept can also be extended into complex three-dimensional (3D) lattices, which can occupy volumes of arbitrary shape and accommodate desired fixturing requirements for practical material applications. In addition, because MNNs inherently have numerous layers of nodes, which are analogous to the neurons within ANNs, MNNs behave as deep neural networks that can learn many complex behaviors simultaneously. If an MNN is damaged, cut to occupy an alternate volume, or fixtured differently, then it can relearn previously mastered behaviors and acquire new behaviors as needed with exposure to changing ambient conditions. An application could include MNN aircraft wings (Fig. 1C) that learn to morph their airfoil shape as desired in response to certain wind-loading scenarios such that the aircraft achieves greater efficiency and maneuverability as it accrues flight experience. This work demonstrates the ability of an MNN to learn two different shape-morphing behaviors using two different algorithms. Experimental and simulated studies were performed to determine the effect of lattice size, packing configuration, algorithm type, behavior number, and linear-versus-nonlinear stiffness tunability on MNN learning.

RESULTS

Learning process

MNNs mechanically learn behaviors analogously to how ANNs mathematically map numerical inputs to outputs. To understand the specifics of how MNNs learn, consider the eight-layer-deep 2D MNN lattice of tunable beams packed in a triangular configuration with eight input and output nodes shown in Fig. 1B. The black bars shown at the top and bottom of the lattice represent fixed ground. Suppose that when the input nodes are loaded by equal horizontal forces, shown as red arrows in Fig. 1B, it is desired that the output nodes respond by moving to target displacements along the contour of the red sinusoidal curve shown in Fig. 1B. To learn this behavior in the midst of unexpected and changing loading scenarios, each tunable beam in the lattice would be prescribed with a random stiffness value. Sensors (e.g., strain gauges on each beam) would then determine the displacement of each node in the lattice for each loading scenario. Because the beam stiffness values and the node displacements are known (i.e., prescribed and measured, respectively), the MNN could determine when the lattice has been loaded with the desired behavior′s loading scenario (i.e., the horizontal forces shown in Fig. 1B). Anytime the desired loading scenario occurs, the lattice sensors would measure the resulting displacements of the output nodes on the lattice′s right side, and the mean squared error (MSE) of these displacements would be calculated by subtracting them from the target displacements and averaging the resulting differences squared. The tunable beams would then change their stiffness values according to an optimization algorithm such that when the process of loading, measuring, and calculating the MSE is repeated, the MSE is minimized until a working combination of beam stiffness values is identified. One possible

combination of beam stiffness values that achieve the desired behavior of Fig. 1B is shown on the left side of Fig. 1D. Different shades of blue are used to denote different axial stiffness values.

Suppose that it is desired that the MNN learns another behavior in addition to retaining the first behavior shown in Fig. 1B. Specifically, suppose that it is desired that the lattice's output nodes displace to an inverted sinusoidal contour, shown as a green curve on the right side of Fig. 1D, in response to its input nodes being loaded by equal vertical input forces, shown as green arrows, instead. To learn the new behavior while maintaining the ability to simultaneously achieve the first behavior, the lattice of tunable beams would begin with the combination of stiffness values that were found to successfully achieve the first behavior. Then, those stiffness values would be adjusted according to the same optimization algorithm to find a new combination of stiffness values that achieve both behaviors simultaneously. This optimization would be achieved by measuring the displacements of the output nodes in response to loading the material's input nodes with alternating horizontal and vertical forces. A single MSE would be calculated that simultaneously considers the results of both loading scenarios. That cumulative MSE would then be minimized so that a desired combination of beam stiffness values would be identified that successfully produced both the new and original behavior. Note that all the tunable beams are colored with the same shades of blue between the two corresponding lattice images of Fig. 1D because a single combination of stiffness values was identified that could successfully enable the MNN to achieve both behaviors.

Because MNNs typically have multiple layers, they can learn the same set of desired behaviors using many different combinations of beam stiffness values. Note that although solution 2 of Fig. 1E exhibits the same desired behaviors as solution 1 of Fig. 1D, it does so with a different combination of beam stiffness values. The fact that many different combinations of beam stiffness values can achieve the same behaviors enables MNNs to learn many new behaviors. Moreover, MNNs do not need to be configured, fixtured, or loaded as shown in the example of Fig. 1B to learn. Any combination of nodes within an MNN could be fixed, loaded as an input, and sensed as an output to learn almost any mechanical behaviors desired.

Tunable beams

There are many ways that the stiffness of a beam could be tuned to enable MNN learning. Principles of jamming (*35*) phase changing (*36*), static balancing (*37*), and electrorheology (*38*), among other approaches (*39*), could be used. Approaches that enable beams to continue exhibiting their prescribed stiffness without external influence (e.g., electrical power, magnetic fields, or temperature) are preferable for MNN applications because such networks physically store a kind of mechanical "muscle memory" in their architecture for manifesting the desired behaviors previously learned.

The beams used to demonstrate the concept of MNNs in this work (Fig. 2A), however, were designed to achieve tunable stiffness via closed-loop active control. This approach was chosen so that any desired linear or nonlinear force-displacement responses (including responses with negative stiffness) could be prescribed to each beam to enable a broader study of MNN learning. These actively controlled beams use voice coils and strain gauges to actuate and sense the deformations of flexures, which guide the extensions and contractions of the beams along their axes. Additional photos of such beams (fig. S1A) and their assembled parts (fig. S1B) are provided in the Supplementary Materials. A detailed discussion about their fabrication and function is provided in Materials and Methods. A diagram (fig. S2A) detailing each beam's closed-loop controller (fig. S2A) is also provided in the Supplementary Materials along with examples of the four calibration plots (fig. S2, B to E) that need to be generated by an Instron testing machine to control each beam's axial stiffness. A discussion about the closed-loop controller and its calibration plots is provided in Materials and Methods. The controller was designed so that when it was set to achieve a linear force-displacement response, the axial stiffness of the tunable beam (i.e., the slope of its response) would be the controller's proportional gain, K_p. Figure 2B provides the force-displacement responses of the tunable beam in Fig. 2A as it is controlled using the linear scenario with different K_p values to achieve various positive and negative stiffness values. A video showing the Instron testing machine cycling the beam as it is being actively controlled with different K_p values is provided (movie S1). The maximum and minimum stiffness values that the beam can be controlled to achieve without becoming unstable or exceeding the actuator's force capabilities were measured to be 2.3 and −2 N/mm, respectively, as shown in Fig. 2C.

Experimental demonstration and study

Tunable beams of the kind shown in Fig. 2A were fabricated and assembled (Fig. 3) within a lattice to experimentally demonstrate and study MNN learning. Four additional actuators were used within decoupling flexures (Fig. 3A) to load the MNN's two input nodes with desired in-plane forces, and cameras (Fig. 3B) were used to directly measure the displacements of the two output nodes. A discussion about the MNN's features, fabrication, and control electronics (fig. S3, A and B) is provided in Materials and Methods. Additional photos (fig. S4, A and B), a video (movie S2), and a discussion about how the MNN was calibrated after its beams were assembled in the lattice are provided in the Supplementary Materials. Plots demonstrating the importance of calibrating MNNs are provided in fig. S5.

MNNs that require external sensors (e.g., cameras) to directly measure the displacements of their output nodes cannot learn without being placed in a testing rig, which is not practical for most applications that require in-field learning. Thus, it is important that the same sensors (e.g., strain gauges) that measure and help control the extension and contraction of their corresponding beams be used to also measure the output-node displacements indirectly to demonstrate practical MNN learning. Thus, the cameras mounted to the frame of the MNN in Fig. 3B were used to validate this indirect approach (i.e., the strain-gauge approach) for measuring output-node displacements. The results of this validation (fig. S6, A and B) and a discussion about how the validation was performed are provided in the Supplementary Materials.

The MNN of Fig. 3B was used in conjunction with this strain-gauge approach to demonstrate that a triangular lattice of 21 tunable beams (shown as blue lines in Fig. 4A) could simultaneously learn two different sinusoidal shape-morphing behaviors (shown in red and green in Fig. 4A for behaviors 1 and 2, respectively). Behavior 1 is manifest when the output nodes, labeled nodes 1 and 2 in Fig. 4A, displace to the right and to the left by 0.5 mm, respectively, as the two input nodes are both pushed to the right (as shown by the red arrows) with equal magnitude. Behavior 2 is manifest when

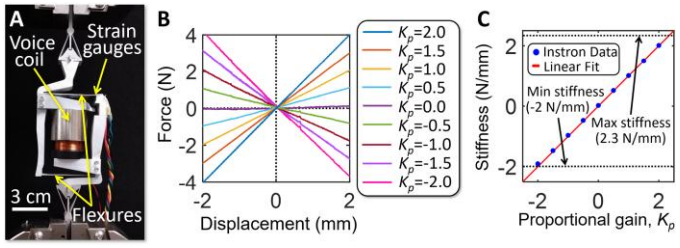

Fig. 2. Tunable beams that use closed-loop control to achieve variable axial stiffnesses. (**A**) Voice coil and strain gauges were used as actuators and sensors to control the axial stiffness of the beam. (**B**) Data collected from an Instron testing machine as it stretches and compresses the tunable beam while it is actively controlled to achieve linear force-displacement responses using different proportional gain values, K_p. (**C**) Plot demonstrating how well the controller's prescribed proportional gain corresponds with the beam's resulting axial stiffness.

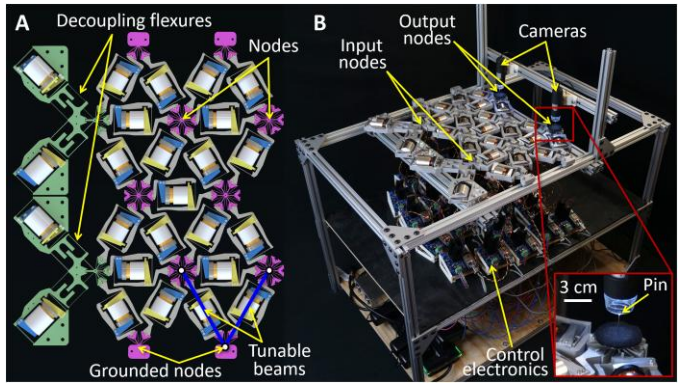

Fig. 3. An MNN. (**A**) A computer-aided design model and (**B**) a photo of the MNN used to conduct the experimental learning study of this work.

nodes 1 and 2 displace in the opposite directions (i.e., to the left and to the right by 0.5 mm, respectively) as the two input nodes are both sheared upward (as shown by the green arrows) with equal magnitude. As the MNN attempted to exhibit these two desired behaviors according to the learning process, the axial stiffness values of each beam were allowed to be tuned between the maximum and minimum values of 2.3 and −2 N/mm, respectively, according to the limits measured in Fig. 2C.

Two optimization algorithms—genetic algorithm (GA) (*40*) and partial pattern search (PPS) (*41*)—were used for learning the two behaviors to compare their performance. The details underlying each optimization algorithm are provided in Materials and Methods, and a video showing the MNN learning is provided in the Supplementary Materials (movie S3). The learning results of the GA and PPS are provided in Fig. 4 (B and C, respectively). The MSE of each algorithm is plotted over time as the MNN learns the two desired behaviors simultaneously, and the initial and final displacements of nodes 1 and 2, relative to the desired target displacements, are also provided for each behavior. A video showing how the nodes move from one displacement to the next (corresponding to each blue dot in the MSE plots of Fig. 4, B and C) is provided in the Supplementary Materials (movie S4).

The MNN of Fig. 4A was also used to compare learning with tunable beams that exhibit linear (e.g., Fig. 2B) versus nonlinear force-displacement responses. Specifically, tangent functions (e.g., the responses shown in fig. S7 for different K_p values) were used for the nonlinear scenario. The MNN's tunable beams were initially set to only exhibit linear force-displacement responses with stiffness values that could vary between 2.3 and −2 N/mm according to the limits measured in Fig. 2C. Two random but different shape-morphing behaviors were generated for the MNN to learn. Each behavior was generated by selecting forces with randomly generated x- and y-axis components between ±2 N, which cause the MNN's output nodes to move selected displacements with randomly generated x- and y-axis components between ±0.5 mm when the selected forces load the input nodes. The MNN then used the PPS algorithm to learn the generated pair of random behaviors simultaneously. The MSE of this learning process over time was recorded, similar to the example plots shown in Fig. 4 (B and C). Five additional random but unique pairs of behaviors were then generated and learned independently by the MNN. The six total resulting MSE-versus-time plots were averaged to produce the single solid-line plot of Fig. 4D (i.e., the plot corresponding to the linear scenario). The same six pairs of generated behaviors were then learned by the same MNN, but its tunable beams were set to only exhibit tangent force-displacement responses (i.e., a nonlinear response) with instantaneous stiffness values that could vary between 2.3 and −2 N/mm according to the limits measured in Fig. 2C. Note that although 2.3 N/mm was found to be the largest axial stiffness value achievable by the tunable beams of this study, that finding is conservative and is only true for instantaneous stiffness values (i.e., beam stiffness values before deformation). When the beam is deformed an appreciable amount, it can be stably controlled with larger stiffness values to accommodate the rising tangent function profile. The six resulting MSE-versus-time plots were averaged to produce the single dotted-line plot of Fig. 4D (i.e., the plot corresponding to the nonlinear scenario).

Before the successful demonstration of the MNN of Fig. 4, different beam designs (fig. S8, A and B) were fabricated and integrated within other MNNs (fig. S9, A to C), which did not successfully learn desired behaviors (fig. S9D). The reasons they failed are discussed in the Supplementary Materials to highlight factors important to successful MNN learning, such as minimal hysteresis (fig. S10), quality sensors, and well-designed flexures.

Simulation study

A computational tool, informed by the measured and modeled (fig. S11) characteristics of the tunable beam of Fig. 2A, was created and used to simulate MNN learning scenarios that the physical MNN of Fig. 3B was not designed to attempt. The tool's assumptions are detailed in the Supplementary Materials, and a discussion about how the tool was verified using finite element analysis (FEA) (fig. S12, A to E) is provided in Materials and Methods. The computational tool was used to generate the example of Fig. 1 (D and E) according to the details also provided in Materials and Methods.

Three simulation studies were conducted using the tool. The MNNs of the first simulation study were all configured as triangular lattices (e.g., Fig. 1B) with eight input and eight output nodes. Their tunable beams were assigned axial stiffness values between 4 and −2 N/mm. Learning was simulated using different numbers of layers and different numbers of random behaviors. Random behaviors were generated by selecting input-node forces and output-node displacements with randomly generated x- and y-axis components between ±1 N and ± 0.5 mm, respectively. To ensure that each

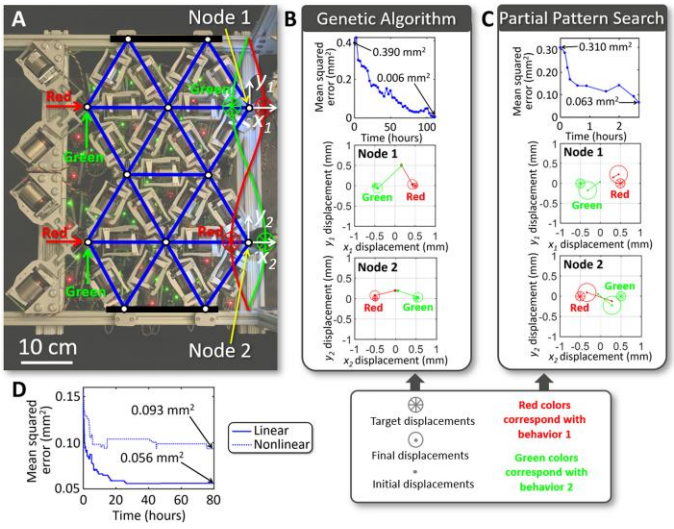

Fig. 4. Experimental study results. (**A**) Two behaviors (shown in red and green) that the MNN attempted to learn using two different optimization algorithms. The results of (**B**) the GA and (**C**) PPS showing MSE over time and the initial and final displacements of the output nodes (i.e., nodes 1 and 2) relative to their target displacements. (**D**) The MNN's MSE plotted over time as its beams were controlled to exhibit tunable linear and nonlinear force-displacement responses.

new behavior generated was sufficiently different from all previously generated behaviors, we calculated an MSE for each previous behavior by averaging the difference between the previous and new behavior's input forces squared. As long as the MSEs that were calculated from each of the previously generated behaviors all exceeded 0.3 N^2, the new behavior was deemed sufficiently different. Once sufficiently different behaviors were generated, three additional unique sets of different behaviors were generated for each scenario. The simulated MNN then attempted to simultaneously learn each unique set of behaviors four times, and the final MSE (i.e., the last MSE that the optimization algorithm achieved by comparing the output-node displacements with the target displacements as described previously) of the attempt that yielded the lowest value was averaged with the lowest final MSEs generated by learning the other unique sets of behaviors. The resulting MSE average was plotted for different numbers of layers and behaviors in Fig. 5A. Ten example behaviors that were randomly generated for this study are shown in fig. S13.

The MNNs of the second simulation study were all configured as triangular lattices (e.g., Fig. 1B) with tunable beams that were assigned axial stiffness values between 4 and −2 N/mm. Learning was simulated using different numbers of layers and different numbers of output nodes (note that the number of output nodes is equal to the number of input nodes). Regardless of the scenario, each MNN attempted to learn the same two behaviors shown in Fig. 1 (D and E), except that the amplitudes of both behaviors' sinusoidal contours on which their target displacements lie were set to 2.5 mm and the shearing input-node forces of the second behavior pushed downward instead of upward with a magnitude of 1 N. Each scenario was attempted 15 times, and the MSE of the simulation that produced the lowest final value was plotted in Fig. 5B.

The MNNs of the third simulation study had eight input and eight output nodes. Their tunable beams were assigned axial stiffness values between 4 and −2 N/mm. Learning was simulated using both triangular and square lattices (Fig. 5C) that learn different numbers of random behaviors with two, four, and eight layers. Random behaviors were generated the same way as in the first simulated study. Once sufficiently different behaviors were generated, three additional unique sets of different behaviors were generated for each scenario. The simulated MNN then attempted to simultaneously learn each unique set of behaviors four times, and the final MSE of the attempt that yielded the lowest value was averaged with the lowest final MSEs generated by learning the other unique sets of behaviors. The resulting MSE average was plotted in Fig. 5C for the different triangular and square lattice scenarios (shown as green and red lines, respectively, in Fig. 5C). Dotted lines correspond to two layers, dashed lines correspond to four layers, and solid lines correspond to eight layers.

DISCUSSION

The concept of MNNs as architected materials that learn behaviors was introduced and experimentally demonstrated using two optimization algorithms, GA and PPS (Fig. 4, B and C). Although the GA proved to be more than 41 times slower than PPS (i.e., GA required 111.13 hours, whereas PPS required 2.68 hours), the MNN of Fig. 3 learned its behaviors 10.5 times more accurately using the GA (i.e., GA achieved an MSE of 0.006 mm^2, whereas PPS achieved 0.063 mm^2). An experimental study was also conducted to compare the learning capabilities of MNNs consisting of tunable beams that are controlled to exhibit linear versus nonlinear stiffness. Figure 4D indicates that MNNs with linear stiffness beams learn with greater accuracy than MNNs with nonlinear stiffness beams (i.e., the lowest linear and nonlinear MSE were 0.056 and 0.093 mm^2, respectively). A computational tool was also created to simulate the effects of lattice size, behavior number, and packing configuration on MNN learning. The plots of Fig. 5 (A and C) demonstrate that the more layers an MNN has and the fewer random behaviors it is tasked to simultaneously learn, the more accurately it can learn (i.e., the lower its final MSE can become). The plot of Fig. 5B demonstrates that as long as an MNN is three or more layers deep, it has enough tunable beams to accurately learn two shape-morphing behaviors regardless of the number of layers and output nodes. Note that although an MNN with fewer output nodes has fewer tunable beams with which to learn, it also has fewer force input and displacement output requirements for the beams to satisfy during learning. Thus, the number of output nodes is largely irrelevant. The plot of Fig. 5C demonstrates that triangular lattices can learn more accurately than square lattices because more tunable beams constitute triangular lattices than square lattices given the same number of layers and output nodes. Moreover, the beams of triangular lattices can more effectively propagate displacements in all directions rather than predominantly along orthogonal directions like square lattices.

MATERIALS AND METHODS

Tunable beam fabrication and function

A photo of a tunable beam used within the MNN of this work is provided in fig. S1A. Its parts are shown disassembled in fig. S1B. It consisted of a Moticont linear voice coil motor (LCVM-032-025-02) actuator and two Hottinger-Baldwin-Messtechnik-GmbH

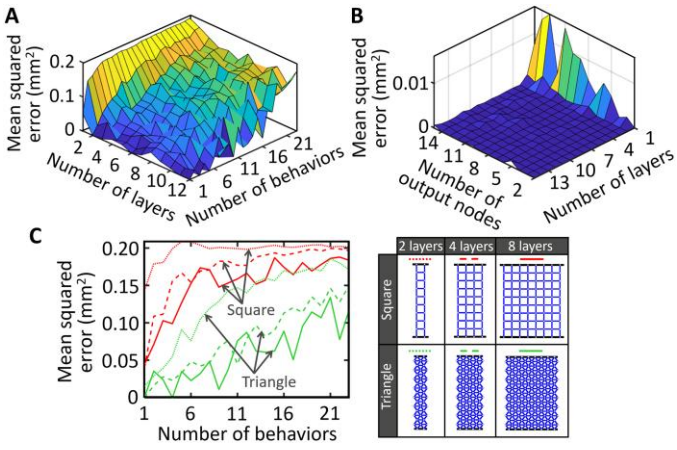

Fig. 5. Simulation study results. The lowest MSE achieved when (**A**) MNNs with different numbers of layers learn different numbers of random behaviors, (**B**) MNNs of different numbers of layers and output nodes learn the same two behaviors, and (**C**) MNNs of different configurations (i.e., triangular and square) learn different numbers of random behaviors.

(HBM) strain gauge (1-LM13-1.5/350GE) sensors, which were mounted on opposing sides near the base of one of the flexure bearings to enhance sensor sensitivity via a Wheatstone circuit in a half-bridge configuration. The rest of the parts were either cut using wire electrical discharge machining (EDM) from 6061-T6 aluminum or, in the case of the brackets, machined from the same material. The two parallel blade flexures behave as linear bearings in that they guide a translational motion along the beam's axis while constraining all other directions. As the flexure bearings deform over their full range, however, they manifest a slight arching parasitic motion, which was considered in the selection and mounting of the voice coil actuator. Care was taken to make sure that the coil portion of the actuator could never make contact with or rub against the outer magnet portion of the actuator so that noise, friction, and hysteresis would be avoided. The brackets were mounted to the beam's body via bolts, and a hard stop (labeled in fig. S1A) was cut into the body to prevent the flexure bearings from yielding by preventing them from deforming beyond ±2.5 mm in either direction. The beam's body can attach to the modular node parts, labeled in fig. S1B, via slide-on dove-tail joints, which were then locked in place by opposing wedges that are pressed together. The resulting joint effectively fused the beam's body to the modular node parts, thus preventing slip-induced friction and hysteresis while also allowing the body to be disassembled and reassembled quickly for debugging or calibration purposes. The utility of this feature is more clearly recognized in the context of the full MNN lattice. Each modular node part used two angled blade flexures to permit rotational deformations about the axis where the planes of the blade flexures intersect (i.e., at the center of the small cylinder shown) while constraining deformations in all other directions. A hard stop (fig. S1B) was used to prevent excessive rotational deformations that would yield the blade flexures. Note that although the tunable beams used to demonstrate the concept of MNNs in this work (fig. S1A) were designed such that only their axial stiffness could be changed, beams that can have their stiffness independently tuned along multiple directions (e.g., axial, transverse, and bending) would likely enhance MNN learning further.

Tunable beam closed-loop controller

The closed-loop control diagram detailing how each beam within the MNN of this work uses proportional-derivative control to achieve tunable axial stiffness is provided in fig. S2A. The digital displacement signal, $e[k]$, is the difference between a reference offset value, r, and the digital displacement feedback signal, $w_D[k]$. The derivative of $e[k]$ is a velocity signal, $v[k]$, which is multiplied by the controller's derivative gain, K_d, which behaves as a damping coefficient. For the purposes of this work, K_d was set to 650. The function, $f(e[k])$, can be set to determine the profile of the tunable beam's force-displacement response. Note that if $f(e[k])$ is set equal to $e[k]$, the beam's force-displacement response will be linear, but if it is set equal to $\tan(e[k])$, it will be a nonlinear tangent function. The output of $f(e[k])$, labeled as $x[k]$ in fig. S2A, is multiplied by the controller's proportional gain, K_p. This proportional gain is set to equal the instantaneous axial stiffness of the beam (i.e., the stiffness of the beam before it is deformed). Note that the K_p values corresponding to each tunable beam within an MNN lattice are the variables that are adjusted during the learning process. Four calibration plots must be generated for each tunable beam in the lattice so that analytical functions can be fit to the measured data collected from an Instron testing machine and used within the control diagram. An example of the first calibration plot is provided in fig. S2B. This plot, called flexure force $g(e[k])$, relates the extension or contraction of the tunable beam along its axis to the force required to deform the beam without control (i.e., the force-displacement response of the passive flexure bearings, labeled in fig. S1B). The force, $F[k]$ (fig. S2A), represents the required voice coil output force to control the beam's axial stiffness as desired. Both this force and $e[k]$ are fed into the second calibration plot (fig. S2C), called voice coil calibration $q(e[k], F[k])$, to generate a force, $F_D[k]$, that corrects for the nonlinearity of the voice coil actuator by multiplying $F[k]$ with a motor scale factor. The sign of $F[k]$ determined whether the pushing or pulling analytical fit function was used. The third calibration plot (fig. S2D), called digital-to-analog converter (DAC) $b(F_D[k])$, converts $F_D[k]$ into a voltage, $V_{FD}(t)$, that is fed to the voice coil actuator within the system's plant (i.e., the tunable beam). The beam responds by displacing an amount, $w(t)$, that causes the strain gauge sensors to produce a voltage, $V_w(t)$, which is then converted into $w_D[k]$ by the fourth calibration plot (fig. S2E), called analog-to-digital converter $h(V_w[k])$.

MNN features, fabrication, and control electronics

The MNN of Fig. 3B consists of 21 tunable beams, which were joined together at nodes (colored purple in Fig. 3A). Each node consists of blade flexures, which permit rotational deformations at each node's center and thereby allow the MNN's lattice to freely deform as it is loaded. The MNN's two input nodes (Fig. 3B) are each loaded by a pair of voice coil actuators that collectively allow their corresponding input node to be loaded with a force that points in any direction within the lattice's plane. These actuators are fixtured within decoupling flexures (shown in green in Fig. 3A) that enable each input node to displace appreciable amounts without imparting transverse jamming loads on the actuators themselves. Hard stops were provided to prevent any flexures within the lattice from yielding as they deformed. Two pairs of grounded nodes (Fig. 3A) are fixtured to a frame along the top and bottom of the two-layer-deep MNN as shown in Fig. 3B. Two cameras

were mounted on the same frame to directly measure the displacement of the two output nodes by tracking pins inserted at their center. Black felt was used to contrast the white color of the pin heads so that they stand out (Fig. 3B).

In addition to the tunable beam parts (fig. S1B) discussed previously, the MNN of Fig. 3B consists of other parts, which were also cut from 6061-T6 aluminum using wire EDM (i.e., the nodes, grounded nodes, and decoupling flexures labeled in Fig. 3A). The four input actuators fixtured within the decoupling flexures are Moticont linear voice coil motors (LVCM-038-038-02), and the two cameras (i.e., Adafruit 636 Digital Microscopes) were mounted to an 80/20 T-slot aluminum frame using parts additively fabricated from acrylonitrile butadiene styrene with a Stratasys UPrint SE Plus 3D printer. Wooden boards were used to support the electronics underneath the MNN.

A close-up photo of the electronics used to control the MNN of Fig. 3B is provided in fig. S3A, and a schematic of the circuit used to drive each tunable beam (i.e., all 21) and each input actuator (i.e., all four) within the MNN is labeled in fig. S3B. Within the circuit, which is current controlled, a Microchip Technology MCP4725 DAC produces a voltage proportional to the desired actuator current, which is supplied to the noninverting input of a Texas Instruments OPA549 operational amplifier. The OPA549 operates in an arrangement similar to a voltage follower, and its output current passes through both the actuator and a Vishay RN55C3500BB14 shunt resistor before reaching the ground. The voltage drop across the shunt resistor is amplified by an Analog Devices AD8226A instrumentation amplifier, which acts as a current-sense amplifier. The output of the AD8226A is in the same range as the DAC output, which is provided to the inverting input of the OPA549 operational amplifier for closed-loop control. To measure the displacement of the tunable beams, the circuit board has another AD8226A instrumentation amplifier, which acts as a strain gauge amplifier. An Espressif ESP32 microcontroller was used to set the DAC input voltage, read the strain gauge voltage, and shut down the OPA549. Stable supply voltages for the analog components were created using a Texas Instruments LM317 voltage regulator for the +12 V supply and an ON Semiconductor MC79M12 voltage regulator for the −12 V supply. Another LM317 voltage regulator was used for the strain gauge supply.

Optimization algorithm details

Optimization algorithms determine how combinations of stiffness values should be assigned to the tunable beams within an MNN for each loading scenario during the learning process. This work used two optimization algorithms to train the MNN of Fig. 4A such that it learned the two shape-morphing behaviors detailed previously. The two optimization algorithms used were a GA and PPS.

The GA used for this work attempts 1000 combinations of axial stiffness values per generation. The most promising combinations (i.e., those that were measured having the lowest MSE) from each generation were then crossed according to MATLAB's "ga" function to generate a new generation of 1000 new combinations of axial stiffness values. The best combination of axial stiffness values (i.e., the one that is measured as having the lowest MSE) from each generation was plotted and corresponds with each blue dot in the uppermost plot of Fig. 4B. The algorithm continued until new generations failed to produce combinations of axial stiffness values with lower MSEs, at which point the algorithm terminated. Note that the uppermost plot of Fig. 4B resulted from 40 generations. Although the GA used for this work requires substantial time and computational power to complete, the algorithm is very thorough and thus produces accurate results.

The PPS algorithm used for this work began with all the tunable beams starting with the same stiffness value (i.e., 1.15 N/mm). A beam was randomly selected, and its currently assigned stiffness value was added to and subtracted from a stiffness increment, which began at 2.15 N/mm. If the two resulting combinations of stiffness values did not reduce the measured MSE, then a different beam was randomly selected, and the same process was repeated. If all the beams in the MNN were subjected to this process and the MSE never reduced for any of them, then the current stiffness increment was multiplied by a reduction factor of 0.9, and the entire process repeated with the new, now smaller, stiffness increment. If adding or subtracting the stiffness increment to the current stiffness value assigned to any beam ever exceeded or fell below the stiffness limit achievable by the beam (i.e., 2.3 and −2 N/mm, respectively, according to Fig. 2C), the beam was assigned the stiffness limit that was surpassed. When a combination of stiffness values was identified that produced a measured reduction in the MNN's MSE, the entire process began again until the current stiffness increment was reduced below a specified threshold (i.e., 0.5 N/mm). Note from the uppermost plot of Fig. 4C that each blue dot corresponds to an event where the MNN's MSE was measured as being reduced, which, for the specific learning example of Fig. 4C, occurred 10 times until the algorithm terminated. Although PPS produced results that are not as accurate as those of the GA used for this work, it required substantially less time and computational power.

Note that despite the fact that both algorithms were designed to identify combinations of stiffness values that produce progressively lower MSEs, the MSEs corresponding to some of the blue dots in the uppermost plots of Fig. 4 (B and C) increase in value compared with prior dots. These temporary increases in plotted MSE values are a result of system noise in the MNN (e.g., sensor noise). Last, note that before nodes 1 and 2 were ever displaced during learning using either algorithm, both output nodes began at the origin of the middle and lowermost plots of Fig. 4 (B and C).

Computational tool verification

The computational tool used to generate the simulation study of this work was verified using FEA. The passive nonaxial stiffness values (i.e., K_1, K_2, and K_3, defined previously) of every tunable beam used within the computational tool's simulation studies were informed via FEA performed on the tunable beam design of fig. S1A without its voice coil actuator or brackets (fig. S11), as discussed previously. The passive axial stiffness value of the same beam design (i.e., its axial stiffness without active control) was also calculated using FEA for 6061-T6 aluminum properties and was found to be 1.81 N/mm. This value was also used to inform the computational tool's tunable beams for the purposes of the tool's verification. A computer-aided design model of the 21-beam MNN lattice of Fig. 4A without its voice coil actuators or brackets is shown in fig. S12A. FEA was performed on this model using the linear deformations and linear material properties of 6061-T6 aluminum to computationally compare various loading conditions with the same loading conditions applied to the same 21-beam lattice simulated by the computational tool. Twenty-five different force combination

attempts, each with x and y components that were selected randomly between ±1 N, were applied to the two input nodes, and the resulting displacements of the two output nodes (i.e., nodes 1 and 2) were calculated and plotted in fig. S12 (B to E) using both FEA and the computational tool of this work. Note that x_1 and y_1 are the displacements of node 1 as measured relative to the origin of x_1 and y_1, labeled in fig. S12A, which is located where node 1 is before lattice deformation. In addition, note that x_2 and y_2 are the displacements of node 2 as measured relative to the origin of x_2 and y_2, labeled in fig. S12A, which is located where node 2 is before lattice deformation. The first force combination attempt shown in the plots of fig. S12 (B to E) corresponds to the FEA results shown in fig. S12A. The fact that the results of the 25 different force combination attempts generated by both the computational tool and FEA correspond well verifies the computational tool's ability to accurately predict the response of general lattice configurations and sizes when subjected to general loading scenarios.

How the computational tool generated the example of Fig. 1 (D and E)

The computational tool generated the learning results of the MNN lattice shown in Fig. 1 (D and E). The amplitudes of both behaviors' sinusoidal contours on which their target displacements lie were set to 2 mm. With a final scaled force magnitude of 0.5 N applied to every input force of both behaviors, solution 1 (Fig. 1D) and solution 2 (Fig. 1E) achieved both behaviors with an MSE of 0.0047 and 0.0008 mm^2, respectively. Note that the nodal displacements of both solutions are all shown with an exaggeration factor of 25 to visually enhance the lattice's behavior. Moreover, although the lattice's tunable beams were subjected to axial, shearing, and bending deformations, which were taken into account during the calculations performed by the computational tool, the deformed beams shown in Fig. 1 (D and E) are graphically depicted as overly simplified straight lines, which directly join the final node locations together.

Supplementary Materials

This PDF file includes:
Materials and Methods
Figs. S1 to S13

Other Supplementary Material for this manuscript includes the following:
Movies S1 to S4

REFERENCES AND NOTES

1. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. *Nature* **521**, 436–444 (2015).
2. A. N. Tait, T. F. de Lima, E. Zhou, A. X. Wu, M. A. Nahmias, B. J. Shastri, P. R. Prucnal, Neuromorphic photonic networks using silicon photonic weight banks. *Sci. Rep.* **7**, 7430 (2017).
3. H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, D. Li, Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE Trans. Industr. Inform.* **14**, 4224–4231 (2018).
4. W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biol.* **5**, 115–133 (1943).
5. F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65**, 386–408 (1958).
6. O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, U. Gana, M. U. Kiru, Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access* **7**, 158820–158846 (2019).
7. K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**, 359–366 (1989).
8. M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, D. B. Strukov, Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**, 61–64 (2015).
9. H.-T. Zhang, T. J. Park, A. N. M. N. Islam, D. S. J. Tran, S. Manna, Q. Wang, S. Mondal, H. Yu, S. Banik, S. Cheng, H. Zhou, S. Gamage, S. Mahapatra, Y. Zhu, Y. Abate, N. Jiang, S. K. R. S. Sankaranarayanan, A. Sengupta, C. Teuscher, S. Ramanathan, Reconfigurable perovskite nickelate electronics for artificial intelligence. *Science* **375**, 533–539 (2022).
10. S. Lee, H. Kim, S.-T. Lee, B.-G. Park, J.-H. Lee, SiO2 fin-based flash synaptic cells in AND array architecture for binary neural networks. *IEEE Electron Device Lett.* **43**, 142–145 (2022).
11. R. Han, P. Huang, Y. Xiang, C. Liu, Z. Dong, Z. Su, Y. Liu, L. Liu, X. Liu, J. Kang, A novel convolution computing paradigm based on NOR flash array with high computing speed and energy efficiency. *IEEE Trans. Circuits Syst. I: Reg. Papers.* **66**, 1692–1703 (2019).
12. S. Dillavou, M. Stern, A. J. Liu, D. J. Durian, Demonstration of decentralized, physics-driven learning. arXiv:2108.00275 [cond-mat.dis-nn] (2022).
13. Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund, M. Soljačić, Deep learning with coherent nanophotonic circuits. *Nat. Photon.* **11**, 441–446 (2017).
14. H. Zhang, M. Gu, X. D. Jiang, J. Thompson, H. Cai, S. Paesani, R. Santagati, A. Laing, Y. Zhang, M. H. Yung, Y. Z. Shi, F. K. Muhammad, G. Q. Lo, X. S. Luo, B. Dong, D. L. Kwong, L. C. Kwek, A. Q. Liu, An optical neural chip for implementing complex-valued neural network. *Nat. Commun.* **12**, 457 (2021).
15. X. Lin, Y. Rivenson, N. T. Yardimci, M. Veli, Y. Luo, M. Jarrahi, A. Ozcan, All-optical machine learning using diffractive deep neural networks. *Science* **361**, 1004–1008 (2018).
16. Z. Wu, M. Zhou, E. Khoram, B. Liu, Z. Yu, Neuromorphic metasurface. *Photon. Res.* **8**, 46–50 (2020).
17. G. Furuhata, T. Niiyama, S. Sunada, Physical deep learning based on optimal control of dynamical systems. *Phys. Rev. Appl.* **15**, 034092 (2021).
18. R. M. Füchslin, A. Dzyakanchuk, D. Flumini, H. Hauser, K. J. Hunt, R. H. Luchsinger, B. Reller, S. Scheidegger, R. Walker, Morphological computation and morphological control: Steps toward a formal theory and applications. *Artif. Life* **19**, 9–34 (2013).
19. S. Boyd, L. Chua, Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE Transactions on Circuits and Systems* **32**, 1150–1161 (1985). 10. 1109/TCS.1985.1085649.
20. W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).
21. H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, W. Maass, Towards a theoretical foundation for morphological computation with compliant bodies. *Biol. Cybern.* **105**, 355–370 (2011).
22. J. C. Coulombe, M. C. A. York, J. Sylvestre, Computing with networks of nonlinear mechanical oscillators. *PLOS ONE* **12**, e0178663 (2017).
23. K. K. Caluwaerts, M. D'Haene, D. Verstraeten, B. Schrauwen, Locomotion without a brain: Physical reservoir computing in tensegrity structures. *Artif. Life* **19**, 35–66 (2013).
24. K. Caluwaerts, J. Despraz, A. Işçen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, V. SunSpiral, Design and control of compliant tensegrity robots through simulation and hardware validation. *J. R. Soc. Interface* **11**, 20140520 (2014).
25. H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, W. Maass, The role of feedback in morphological computation with compliant bodies. *Biol. Cybern.* **106**, 595–613 (2012).
26. K. Nakajima, H. Hauser, T. Li, R. Pfeifer, Information processing via physical soft body. *Sci. Rep.* **5**, 10487 (2015).
27. M. Hermans, M. Burm, T. V. Vaerenbergh, J. Dambre, P. Bienstman, Trainable hardware for dynamical computing using error backpropagation through physical media. *Nat. Commun.* **6**, 6729 (2015).
28. L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, P. L. McMahon, Deep physical neural networks trained with backpropagation. *Nature* **601**, 549–555 (2022).
29. T. A. Schaedler, W. B. Carter, Architected cellular materials. *Annu. Rev. Mat. Res.* **46**, 187–210 (2016).
30. F. Zangeneh-Nejad, D. L. Sounas, A. Alù, R. Fleury, Analogue computing with metamaterials. *Nat. Rev. Mater.* **6**, 207–225 (2021).
31. S. Zuo, Q. Wei, Y. Tian, Y. Cheng, X. Liu, Acoustic analog computing system based on labyrinthine metasurfaces. *Sci. Rep.* **8**, 10103 (2018).
32. T. W. Hughes, I. A. D. Williamson, M. Minkov, S. Fan, Wave physics as an analog recurrent neural network. *Sci. Adv.* **5**, eaay6946 (2019).
33. M. Stern, D. Hexner, J. W. Rocks, A. J. Liu, Supervised learning in physical networks: From machine learning to learning machines. *Phys. Rev. X.* **11**, 021045 (2021).
34. M. Stern, C. Arinze, L. Perez, S. E. Palmer, A. Murugan, Supervised learning through physical changes in a mechanical system. *Proc. Natl. Acad. Sci. U.S.A* **117**, 14843–14850 (2020).
35. E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. R. Zakin, H. Lipson, Universal robotic gripper based on the jamming of granular material. *Proc. Natl. Acad. Sci. U.S.A* **107**, 18809–18814 (2010).

36. R. Poon, J. B. Hopkins, Phase-changing metamaterial capable of variable stiffness and shape morphing. *Adv. Eng. Mater* **21**, 1900802 (2019).
37. P. R. Kuppens, M. A. Bessa, J. L. Herder, J. B. Hopkins, Monolithic binary stiffness building blocks for mechanical digital machines. *Extreme Mech. Lett* **42**, 101120 (2021).
38. K. Wei, Q. Bai, G. Meng, L. Ye, Vibration characteristics of electrorheological elastomer sandwich beams. *Smart Mater. Struct* **20**, 055012 (2011).
39. L. Blanc, A. Delchambre, P. Lambert, Flexible medical devices: Review of controllable stiffness solutions. *Actuators* **6**, 23 (2017).
40. D. Whitley, An overview of evolutionary algorithms: Practical issues and common pitfalls. *Inf. Softw. Technol* **43**, 817–831 (2001).
41. E. D. Dolan, R.M. Lewis, V. Torczon, On the local convergence of pattern search. *SIAM J. O.* **14**, 567–583 (2003). 10.1137/S1052623400374495.

Acknowledgments: We thank program officer B. "Les" Lee for generous support. **Funding:** This work was supported by Air Force Office of Scientific Research FA9550-18-1-0459 (to J.B.H.) and Air Force Office of Scientific Research FA9550-22-1-0008 (to J.B.H.). **Author contributions:** Conceptualization: J.B.H. Experimentation: R.L. and J.B.H. Simulation: R.L., E.A.B.M., and J.B.H. Writing: J.B.H. **Competing interests:** The authors are inventors on a provisional patent application (no. 63/369,065, "Mechanical neural-network-based metamaterial that learns its properties") filed by the University of California. **Data and materials availability:** All data and code scripts needed to evaluate the conclusions in the paper are present in the paper or the Supplementary Materials. The data for this study have been deposited on GitHub at https://zenodo.org/badge/latestdoi/482735718.

Submitted 27 April 2022
Accepted 19 September 2022
Published 19 October 2022
10.1126/scirobotics.abq7278

Science Robotics

Mechanical neural networks: Architected materials that learn behaviors

Ryan H. Lee, Erwin A. B. Mulder, and Jonathan B. Hopkins

Sci. Robot. **7** (71), eabq7278. DOI: 10.1126/scirobotics.abq7278

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.