One for all: Universal material model based on minimal state-space neural networks

Colin Bonatti1 and Dirk Mohr1,2*

Computational models describing the mechanical behavior of materials are indispensable when optimizing the stiffness and strength of structures. The use of state-of-the-art models is often limited in engineering practice due to their mathematical complexity, with each material class requiring its own distinct formulation. Here, we develop a recurrent neural network framework for material modeling by introducing “Minimal State Cells.” The framework is successfully applied to datasets representing four distinct classes of materials. It reproduces the three-dimensional stress-strain responses for arbitrary loading paths accurately and replicates the state space of conventional models. The final result is a universal model that is flexible enough to capture the mechanical behavior of any engineering material while providing an interpretable representation of their state.

INTRODUCTION

Before novel materials can find their way into real-life products, their mechanical behavior needs to be understood and modeled. Constitutive material modeling entails predicting the stress as a function of the deformation history applied onto a material. The corresponding algorithmic description of the material response is incremental. The loading history is considered as a sequence of strain increments $|\Delta \epsilon(t_1), \ldots, \Delta \epsilon(t_N)|$. The material model then typically predicts the stress tensor $\sigma(t_n)$ and the state $\chi(t_n)$ of a material at time $t_n$ as a function of the current strain increment tensor $\Delta \epsilon(t_n)$ and the material state $\chi(t_{n-1})$

$$f:|\Delta \epsilon(t_n), \chi(t_{n-1})| \rightarrow |\sigma(t_n), \chi(t_n)|$$

Aside from finding the mapping $f$, a major challenge rests in the choice of the size and contents of the state variable vector $\chi$. This vector describes succinctly the evolution of the material’s microstructure over time. Consequently, its formulation is highly specialized. New classes of materials often require the development of new classes of models with a custom state-space representation (see Fig. 1A). Conceptually, recurrent neural networks (RNNs) (1) could alleviate this issue by delivering a universal material model, capable of modeling any material through a simple change of parameters. RNNs function incrementally and feature memory units, which naturally serve as state variables in the context of material modeling, making them uniquely suited to this problem (2). At each time step, an input function preprocesses the current inputs, a transition function updates the memory units, and an output function converts the memory units into observable outputs (3). The training of RNNs is notoriously difficult (4). Typical recurrent networks rely on long short-term memory (LSTM) or gated recurrent unit (GRU) architectures (5, 6) to mitigate these problems through gated transition functions (see Table 1). However, these transition functions are inherently shallow (7). Consequently, LSTM or GRU models with few state variables are necessarily very small (see Supplementary Text), which limits their representational power. Additional flexibility is obtained by either widening the layers or stacking (8) them (fig. S1), thereby adding memory units to the network. As a result, applications of LSTMs and GRUs to material modeling (9–15) involve far more state variables than typical physics-based models.

From a scientific viewpoint, this excessive use of memory units prevents the identification of interpretable state variables and raises concerns regarding variable redundancy, overfitting, and extrapolation. We postulate that a universal material model should feature a small, physically meaningful number of state variables but allow for parameter-rich transition and output functions to provide flexibility. Instead of adopting models that have been developed for nonmechanical applications such as natural language processing (16), we design an RNN for material modeling with deep transition and output functions that are decoupled from the number of memory units, enabling the design of RNNs with “minimal state space,” that represent the material with a minimal number of meaningful state variables. The transition function of the resulting Minimal State Cell (MSC) features $d_\ell$ (transition depth) quadratic layers $\ell$, of width $w$, followed by LSTM-inspired gates. Quadratic layers are chosen as they are compared favorably to linear layers (see fig. S2). The updated state vector $\chi(t_n)$ of size $n_\ell$ is converted into outputs using $d_o$ (output depth) quadratic layers $\lambda$ of width $w$ (see Fig. 1B and Table 1). Contrary to stacked GRUs and LSTMs, the MSC features a single feedback loop encompassing an arbitrarily wide and deep neural network (compare Fig. 1B with fig. S1).

We illustrate the efficiency and universal modeling capability of the MSC by reproducing four different physics-based material models: (i) an isotropic-hardening elastoplastic model, (ii) a mixed-hardening elastoplastic model, (iii) a crushable foam model, and (iv) a hyperelastic rubber model with internal damage. For any given model, describing the material state requires theoretically six external state variables (typically either the stress or the strain tensor components) plus a material-dependent number of internal state variables. For each material, we use finite element (FE) simulations to generate a training dataset and a validation dataset where the inputs are random sequences of 125 strain increments and the outputs are the corresponding (normalized) stress sequences. We train MSC, LSTM, and GRU networks by minimizing the mean squared error $J_{\text{mse}}$ in the predicted outputs using a custom training strategy and evaluate the networks based on their validation loss $J_{\text{val}}$.  

---

1Department of Mechanical and Process Engineering, Swiss Federal Institute of Technology (ETH), Tannenstrasse 3, Zurich 8006, Switzerland. 2Department of Mechanical Engineering, Massachusetts Institute of Technology (MIT), 77 Massachusetts Avenue, Cambridge, MA 02139, USA.  

*Corresponding author. Email: dmohr@ethz.ch

Bonatti and Mohr, Sci. Adv. 2021; 7 : eabf3658 23 June 2021
RESULTS AND DISCUSSION

In the first experiment, we explore the effect of the state-space size $n_s$. We use the MSC to identify the optimal number of state variables and provide evidence that state variable redundancy is not desirable. For that purpose, we consider MSCs with fixed architecture except for the memory size ($n_s = 1, 2, \ldots, 25$). Similar loss levels (Fig. 2, A to D) on training and validation datasets demonstrate that the training datasets are large enough to prevent overfitting. The obtained $\text{J}_{\text{mse}}$ versus $n_s$ graphs quantify the impact of adding or subtracting state variables to the model. In each case, the loss gradually decreases until a critical state-space size $n_s^*$, after which it follows a plateau. We obtain $n_s^* = 12$ for the mixed-hardening material and $n_s^* = 7$ for the others, which matches the number of state variables of the corresponding physics-based models. In other words, the MSC allows us to identify the optimal size of the material’s state space. The observed plateaus prove empirically that adding state variables does not increase the accuracy of the model. Closer inspection reveals that models with excess state variables spread the relevant information over all available memory units (see fig. S3).

Before developing MSCs with self-architecting state spaces, we can already obtain models with truly minimal representations of the material’s state space by manually selecting the memory size equal to $n_s^*$. For comparison, we perform the same analysis for LSTM and GRU networks on the isotropic-hardening and mixed-hardening materials. Because of the inherent coupling between state-space size and number of parameters for these architectures (see Supplementary Text), the networks are necessarily single-layered and small—too small to represent the desired material behavior accurately (Fig. 2, A and B). Consequently, these architectures allow neither for the identification of a minimal state-space size nor for the accurate modeling of the stress-strain response when they are used in a minimal state space configuration.

In a second experiment, we explore the effect of the deep transition function architecture and show that optimal state-space MSCs perform far better for the same number of parameters than state-of-the-art stacked LSTM and GRU models. Focusing our attention...
Table 1. Constitutive equations of RNNs. Mathematical details of incremental update \( \Delta \epsilon(t_n), \chi(t_{n-1}) \rightarrow (\sigma(t_n), \chi(t_n)) \) for a single LSTM cell (left) and an MSC (right). The symbol \( \phi \) designates the logistic sigmoid activation function.

| LSTMs                      | MSC                        |
|----------------------------|----------------------------|
| Input                      |                            |
| \( I_0 = \begin{bmatrix} \Delta \epsilon(t_n) \\ \chi(t_{n-1}) \end{bmatrix} \) & \( I_0 = \begin{bmatrix} \Delta \epsilon(t_n) \\ \tanh(\chi(t_{n-1})) \end{bmatrix} \) |
| (internal)                 |                            |
| \( g = \phi(W_i \cdot x_n + b_i) \) & \( g = \phi(W_i \cdot x_n + b_i) \) |
| \( g_i = \phi(W_i \cdot x_n + b_i) \) & \( g_i = \phi(W_i \cdot x_n + b_i) \) |
| \( g_0 = \phi(W_0 \cdot x_n + b_0) \) & \( g_0 = \phi(W_0 \cdot x_n + b_0) \) |
| Transition                 |                            |
| \( d\chi = \tanh(W_\chi \cdot I_n + b_\chi) \) & \( d\chi = \tanh(W_\chi \cdot I_n + b_\chi) \) |
| (update)                  |                            |
| \( \chi(t_n) = g_i \cdot \chi(t_{n-1}) + g_f \cdot d\chi \) & \( \chi(t_n) = g_i \cdot \chi(t_{n-1}) + g_f \cdot d\chi \) |
| Output                     |                            |
| (internal)                 |                            |
| \( \sigma(t_n) = W_y \cdot \chi(t_n) + b_y \) & \( \sigma(t_n) = W_y \cdot \chi(t_n) + b_y \) |
| (linear)                  |                            |

Fig. 2. Effect of model architecture on the performance of MSCs. (A to D) Effect of the number of state variables \( n_s \) of MSCs, with comparison (A and B) to single-layer GRUs and LSTMs. (A) Isotropic-hardening model (7 state variables). (B) Mixed-hardening model (12 state variables). (C) Foam model (7 state variables). (D) Rubber model (7 state variables). (E and F) Effect of the number of parameters on the mean squared error on the validation datasets for optimal state-space MSCs compared with stacked LSTMs and GRUs for the (E) isotropic-hardening model and (F) mixed-hardening model.

on the isotropic-hardening and mixed-hardening models, we consider linear output\((d_o = 0)\) MSCs of various widths \( w \) and transition depths \( d_t \) and stacked LSTMs and GRUs of the same widths and depths. Comparing the validation losses leads to similar conclusions for both materials (Fig. 2, E and F): For all network types, the performance increases with the number of parameters, while deep\((d \geq 3)\) networks perform better than shallower networks of similar size. We observe no notable difference between stacked LSTMs and GRUs. However, both architectures perform substantially poorer than similar-sized MSCs. For the isotropic-hardening
model, deep MSCs with about 12,000 parameters offer equivalent accuracy to the largest stacked models (more than 800,000 parameters) while being more than 10 times faster.

In the third series of experiments, we elucidate the predictive capabilities of a relatively compact MSC (5864 parameters) trained on the mixed-hardening model. We show that the predicted solutions lose accuracy for small strain increments and provide a solution to this issue. Given that explicit FE solvers typically require very large numbers of small increments, we include test datasets with 500- and 100,000-long increment sequences but similar arc

![Diagram](https://example.com/diagram.png)

**Fig. 3. Predictive capabilities of MSCs and learned features for different materials.** Results for compact MSC models (with $d_t = 4$ and $w = 25$) after training with random walk data compared to Finite Element Model (FEM) results. (A to C) First stress component prediction on the worst predicted example for the mixed-hardening material: (A) validation dataset, (B) 500-long test dataset, and (C) 100,000-long test dataset. (D) Bauschinger effect for cyclic loading of the mixed-hardening material. (E) Mullins effect during unloading-reloading of the rubber material. (F and G) Spearman correlation coefficients between state variables of MSCs and state variables of the physics-based models evaluated on the validation dataset of (F) the isotropic-hardening model and (G) the crushable foam model. (H and I) Internal state variable of an MSC model as a function of the equivalent plastic strain of the corresponding physics-based model for the 500 validation examples for the (H) isotropic-hardening model and (I) crushable foam model. (J and K) Extraction of yield surface and demonstration of their self-similar evolution by plotting iso-contours of the MSC-inferred internal state variable. Gray lines indicate the isolines of the theoretical yield surface for comparison. (J) Iso-levels (0.0001, 0.01, 0.02, 0.03, 0.04, and 0.05) obtained from 3000 isochoric proportional loading path simulations for the isotropic-hardening model. (K) Iso-levels (0.003, 0.03, 0.06, 0.09, 0.12, and 0.15) obtained from 1000 proportional loading path simulations for the crushable foam model.
lengths as the training examples. Figure 3 (A to C) presents the first uniaxial stress component of the worst predicted path for each validation and testing dataset. The validation example (Fig. 3A) reveals excellent agreement between the neural network (magenta curves) and physics-based (black curves) models. The MSC also extrapolates well for the 500-step long test set. This good agreement is lost for longer sequences, for which strain increments are substantially smaller than those included during training. In contrast to many standard iterative schemes, the precision of neural network predictions decreases with the strain increment size. To facilitate the application of MSCs in engineering practice, we address this issue by extrapolating the network’s response in a linear manner whenever the strain increment norm falls below a chosen threshold (0.035). The accurate stress predictions provided by the enhanced MSC model on the examples considered (cyan curves) irrespective of the size of the strain increments demonstrate the potential of this linearization approach. However, the large threshold used implies that the stiffness agreement is lost, which indicates the need for stricter evaluation metrics of surrogate models in the future.

In the fourth series of computational experiments, we demonstrate that MSCs successfully learn the characteristic features of the stress-strain response of different classes of materials. We also make a first attempt to interpret MSC-identified state variables. We begin by verifying that the MSC captures the characteristic Bauschinger effect of the mixed-hardening material. For this, we consider a cyclic tension-compression loading with increasing amplitudes. The MSC displays the expected phenomenon (Fig. 3D), despite being trained on random walks. For the rubber model, we consider loading-unloading cycles of increasing amplitude. Here, the MSC correctly reveals (Fig. 3E) the governing damage mechanism (Mullins effect).

We now turn our attention to the foam and isotropic-hardening materials. For each material, we investigate the Spearman (rank) correlation between the state variables of the neural network (predicted for the validation dataset) and the state variable of the corresponding physics-based model. Figure 3 (F and G) reveals that in both cases, the MSCs separate the state variables into three groups: (i) pressure, (ii) deviatoric stresses (five variables), and (iii) one additional internal state variable. Figure 3 (H and I) presents the MSC-inferred internal state variables for both materials (in each case, these correspond to component \( \chi_p \)) as a function of the equivalent plastic strain, i.e., the internal state variable of the corresponding physics-based models. For both materials considered, the MSC-inferred state variable is a monotonic function of the physics-based one. In other words, the MSCs effectively reverse-engineer the state space used by the physics-based models. We conclude by showing iso-contours of the MSC’s internal state variable for proportional strain paths. The resulting self-similar contours in principal stress space (Fig. 3I) predicted by the MSC model are an important feature of the response of von Mises solids with isotropic hardening. The lowest contour (black curve; value, 0.0001) closely follows the theoretical yield surface (within 1%). For the crushable foam, the MSC model predictions display the characteristic self-similar half-ellipses in the pressure versus equivalent stress space (Fig. 3K).

In the fifth and final experiment, we use transfer learning to demonstrate that once an MSC has learned the intrinsic mechanisms for a first material, it can quickly spawn precise models for other members of the same class of materials (as they share the same mechanisms). For illustration, we focus our attention on von Mises solids with isotropic hardening. In addition to the datasets for low carbon steel, we construct datasets for a copper alloy, a second material of this class with different elasticity and hardening properties. Starting with a small model (5194 parameters) trained on the first material, we freeze the 4600 weights for all internal layers and only train the 594 weights of the gates and the output function. The resulting model of copper obtained through transfer learning achieves better results (\( J_{\text{valid}}^{\text{mse}} \approx 9.2 \times 10^{-6} \)) than the models of the same architecture trained directly following the protocol of experiments 1 and 2 (\( J_{\text{valid}}^{\text{mse}} \approx 2.4 \times 10^{-5} \)), in a 10th of the time. These results suggest that after training for a first material, the internal layers have captured the governing mechanisms of von Mises linear-hardening materials as a whole, thereby enabling the rapid identification of models for other materials through transfer learning.

In sum, we have developed MSCs, a deep learning model that addresses the specific needs of material modeling. The outcome is a first universal material model that is able to represent the three-dimensional stress-strain response of a wide range of materials. The MSCs trained herein not only provide accurate stress predictions on random strain paths but also are able to recognize important physical mechanisms such as isotropic hardening, the Bauschinger effect, pressure-dependent yield, or irreversible damage of rubbers. MSCs also replicate the state space of constitutive models in an interpretable manner. The results strongly suggest that MSCs will facilitate the identification of compact representations of the state of complex systems, such as composites and metamaterials, and advance the understanding of their history-dependent properties. The proposed approach has the potential to disrupt modern material modeling not only by unifying models of existing and future materials but also by reaching commercially interesting levels of computational performance via massive parallelization of the neural network operations.

MATERIALS AND METHODS

Physics-based material models

The isotropic-hardening material is defined with simple parameters that mimic the behavior of a low carbon steel. The parameters of the mixed-hardening model (17), the foam model (isotropic hardening) (18), the rubber model (no permanent set) (19), and the copper model (20) used for transfer learning are taken from the Abaqus Documentation. The parameters for the copper model are adapted to eliminate temperature effects. Details are as follows:

1) Isotropic hardening: isotropic elasticity with Young’s modulus of 200 GPa and Poisson’s ratio of 0.3 and von Mises plasticity with a Swift hardening function using \( A = 500 \) MPa, \( n = 0.2 \), and \( \varepsilon_0 = 0.05 \)

\[
K(\bar{\varepsilon}) = A(\varepsilon_0 + \bar{\varepsilon})^n
\]

2) Mixed hardening: isotropic elasticity with Young’s modulus of 210 GPa and Poisson’s ratio of 0.3. Von Mises plasticity with mixed isotropic-kinematic hardening composed of Voce isotropic hardening defined through

\[
K(\bar{\varepsilon}) = \sigma_y + Q(1 - \exp(-\beta \bar{\varepsilon}))
\]

with \( \sigma_y = 200 \) MPa, \( Q = 2 \) GPa, and \( \beta = 0.26 \), and kinematic hardening defined through

\[
\alpha = \frac{C}{\gamma} (\sigma - \alpha) \bar{\varepsilon} - \gamma \alpha \bar{\varepsilon}
\]

where \( \alpha \) is the backstress, \( C = 25.5 \) GPa, and \( \gamma = 81 \).
3) Foam model: isotropic elasticity with Young’s modulus of 3 MPa and Poisson’s ratio of 0.2. Crushable foam plasticity with isotropic hardening, and input parameters the uniaxial compressive yield stress ratio $k = 1.1$ and the plastic Poisson’s ratio $v_p = 0$. Tabulated hardening curve, with initial (uniaxial), yields stress 0.22 MPa [see online repository (21) or Abaqus Documentation (18) for details].

4) Rubber model: Hyperelasticity with Yeoh strain energy potential with reduced parameters

$$U = C_{10}(I_1 - 3) + C_{20}(I_1 - 3)^2 + C_{30}(I_1 - 3)^3 + \frac{1}{2}D_1(J - 1)^2$$

where $U$ is the volumetric strain energy, $I_1$ is the first deviatoric strain invariant, and $J$ is the volume ratio. Parameters are as follows: $C_{10} = 19.1122$ MPa, $C_{20} = 4.9415$ MPa, $C_{30} = 0.5346$ MPa, and $(D_1)^{-1} = 2 \cdot 10^5$ MPa. Mullins effect deduced from tabulated test curves [see online repository (21) or Abaqus Documentation (18)] for details.

5) Copper model: isotropic elasticity with Young’s modulus of 18 GPa and Poisson’s ratio of 0.34, von Mises plasticity with tabulated hardening based on a Johnson-Cook law with $B = 90$ MPa, $C = 292$ MPa, and $n = 0.31$

$$K(\tilde{\varepsilon}^p) = B + C(\tilde{\varepsilon}^p)^n$$

**Random walks in strain space**

We define random strain paths by their number of strain increments $N$, a maximum volume change $\Delta V$, and a smoothening parameter $m$. Logarithmic strain increments $\Delta \varepsilon_{ij}(t)$ with $ij = 11,22,33,12,13,23$, and $t = 1...N$ are first drawn from a uniform distribution

$$\forall t, \forall i, j, \Delta \varepsilon_{ij}(t) \sim U(-0.5,0.5)$$

We then multiply these increments by an envelope term $\delta(t)$ that varies every 10 increments

$$\forall t = 1...N \frac{10}{N} \log_{10}(\delta(t)) \sim U(-3,-1)$$

The corresponding volumetric strain is obtained through

$$\forall t, \Delta \varepsilon_{vol}(t) = \Delta \varepsilon_{11}(t) + \Delta \varepsilon_{22}(t) + \Delta \varepsilon_{33}(t)$$

We then define a target capped volumetric strain path $\varepsilon_{vol}$ [with $\varepsilon_{vol}(0) = 0$] through

$$\forall t, \varepsilon_{vol}(t) = \max\left(-\Delta V, \min(\Delta V, \varepsilon_{vol}(t-1) + \Delta \varepsilon_{vol}(t))\right)$$

Before extracting strain increments

$$\forall t, \Delta \varepsilon_{vol}(t) = \varepsilon_{vol}(t) - \varepsilon_{vol}(t-1)$$

$$\forall i, j, \Delta \varepsilon_{ij}(t) = \Delta \varepsilon_{ij}(t) + \frac{1}{3}(\Delta \varepsilon_{vol}(t) - \Delta \varepsilon_{vol}(t))$$

$$\forall i \neq j, \Delta \varepsilon_{ij}(t) = \Delta \varepsilon_{ij}(t)$$

Last, we smoothen the strain paths with a Gaussian filter $g_m$ of kernel size $m$

$$\forall i, j, \varepsilon_{ij} = \varepsilon_{ij} \ast g_m$$

### Single-element simulations

We perform single-element simulations to evaluate the physics-based constitutive models for random strain paths, using the FE solver Abaqus/Standard. We convert strain paths into nodal displacements applied to a single hexahedral element (C3D8R). From each simulation, we extract the histories of the logarithmic strains $\varepsilon_{ij}(t)$ and Cauchy stresses $\sigma_{ij}(t)$ at the central integration point.

### Dataset definitions

For training and validation datasets, we use $N = 125$ strain increments and $m = 1$ (no smoothening). Training datasets include the results from single-element simulations for 10,000 paths for all materials except the rubber (100,000 examples; see fig. S4 and Supplementary Text). Validation datasets contain 500 paths. For testing datasets, we use 100 paths with $N = 500$ and $m = 50$. We obtain long sequences with 100,000 increments from the same strain paths by requesting finer output sampling. We define a maximum logarithmic volume change $\Delta V = 0.03$, except for the foam ($\Delta V = 2.3$).

Input and output vectors for the MSC model are composed of the strain increments $\Delta \varepsilon_{ij}(t)$ and stresses $\sigma_{ij}(t)$

$$\Delta \varepsilon(t) = [\Delta \varepsilon_{11}(t); 2\Delta \varepsilon_{12}(t); 2\Delta \varepsilon_{13}(t); \Delta \varepsilon_{22}(t); 2\Delta \varepsilon_{23}(t); \Delta \varepsilon_{33}(t)]$$

$$\sigma(t) = \left[\begin{array}{c}
\sigma_{m}(t)\
\sigma_{11}(t)\
\sigma_{22}(t)\
\sigma_{12}(t)\
\sigma_{13}(t)\
\sigma_{23}(t)\
\sigma_{33}(t)
\end{array}\right]$$

where $\sigma_{m}(t)$ is the average stress

$$\sigma_{m}(t) = \frac{\sigma_{11}(t) + \sigma_{22}(t) + \sigma_{33}(t)}{3}$$

where $\sigma_{ij} = \sigma_{ii} - \sigma_{mi}$ are deviatoric stress components, and $p_0$ and $\sigma_0$ are normalizing constants. We take $p_0$ as 5 GPa, 5 GPa, 0.5 MPa, 12 GPa, 4 GPa and $\sigma_0$ as 500, 500, 0.5, 125, and 400 MPa for the isotropic-hardening, mixed-hardening, crushable foam, rubber, and copper (i.e., the transfer learning target) models, respectively.

### Neural network implementation

We implement the MSC and enhanced MSC models as custom recurrent networks with the help of the Keras (22) library for python. We initialize weights using the Keras defaults for LSTMs, with the exception of the vectors $b_{i,j}^0$ and $b_{i,j}^p$ for which we use the initial value 1.

### Training strategy

We train networks on a single GPU using 800-MB memory (3 GB for the rubber model) using Keras (version 2.3.1) with a TensorFlow (version 1.14.0) backend (23). Results were sensitive to the versions used. We use the optimizer Adam (24) with default parameters, a clipnorm parameter of 0.01, and a batch size of 64, in three steps. Noting $n_r$ the number of epochs, we use

1) A pretraining step of $n_r/100$ epochs using only the first 25 time increments of each sequence and learning rate of 0.0005.

2) A main training step of $n_r$ epochs using all time increments and a custom Keras callback function, which modifies the learning rate $\alpha$ (initial value of 0.005) as a function of the loss observed on each batch:

$$\alpha_{new} = \min(\alpha_{old} \cdot 0.1 \cdot \sqrt{\text{mse}_{\text{batch}}})$$
3) A refining phase of $n_c/100$ epochs using a 10 times learning rate scheme.

A custom Keras callback is used to mitigate cases of sudden and sustained loss rise: If a loss 50% higher than the current achieved minimum (over an epoch) is maintained for 20 epochs, the weights corresponding to the minimum loss achieved are restored. Some trainings (about 10%) ended with NaN values and were relaunched.

**First series of numerical experiments**

We train MSCs with $w = 50$, $d_f = 3$, and $n_i = 1, 2, \ldots, 25$ for each material. For the larger rubber dataset, we use $d_f = 2$ (see fig. S4 and Supplementary Text) and $n_i = 500$. For all other materials, we use $d_f = 0$ and $n_i = 2500$. In each case, five repeats are trained and presented.

**Second series of numerical experiments**

We train MSCs with linear output function ($d_f = 0$) and variable widths and depths on the isotropic-hardening and mixed-hardening datasets. We use a grid with $d_f = 1, 2, 3, 4, 5$ and $w = 15, 25, 40, 50, 75, 100, 150$. We compare them to stacked LSTM and GRU models with $d = 1, 2, 3, 4, 5$ stacked layers of width $w = 15, 25, 40, 50, 75, 100, 150$ using the Keras implementation of these layers with default parameters. Five repeats are trained for each architecture with $n_i = 2500$. We compare speeds by evaluating the networks on 10,000 random paths of length 1000 with a batch size of 1000 on a personal computer.

**Third series of numerical experiments**

Ten trainings of MSCs with $d_f = 4$, $w = 25$, and $d_f = 0$ are performed with $n_i = 25,000$. The neural network with the best loss on the validation dataset is presented. We consider the following norm for strain increments

$$\|\Delta \varepsilon\| = \frac{1}{2}(\Delta \varepsilon_{11}^2 + \Delta \varepsilon_{22}^2 + \Delta \varepsilon_{33}^2 + 2\Delta \varepsilon_{12}^2 + 2\Delta \varepsilon_{13}^2 + 2\Delta \varepsilon_{23}^2)$$

Noting $f_i : (\Delta \varepsilon(t_0), \chi(t_{n-1})) \rightarrow \chi(t_n)$ the MSC’s transition function and $\varepsilon_0$ the threshold used, we define a locally linearized transition

$$f_i : (\Delta \varepsilon, \chi) \rightarrow \left\{ \begin{array}{ll} f_i(\Delta \varepsilon, \chi) if \|\Delta \varepsilon\| > \varepsilon_0 \\ \chi + \frac{\varepsilon_0}{\|\Delta \varepsilon\|}(f_i(\varepsilon_0, \chi) - \chi) otherwise. \end{array} \right.$$ 

We implement the resulting enhanced MSC as a custom recurrent layer in Keras.

**Fourth series of numerical experiments**

For the rubber material, we train 10 MSCs with $d_f = 4$, $w = 25$, and $d_f = 2$ using $n_i = 2500$. For other materials, we train 10 MSCs each with $d_f = 4$, $w = 25$, and $d_f = 0$ with $n_i = 25,000$. This difference in $n_i$ compensates for the difference in dataset size—the same number of iterations is used for each model. In each case, we present the neural network with the best loss on the considered dataset.

**Fifth series of numerical experiments**

We use the same model trained on the isotropic material, as in the fourth experiment. We define a new model where the weight matrices $W_{ij}^0$, $W_{ij}^1$, and the bias vectors $b_{ij}^0$ and $b_{ij}^1$ are frozen (i.e., not updated during training) and import the weights of the trained model (4600 frozen parameters and 594 trainable parameters). We train five transferred models with $n_c = 250$ and no pretraining phase. We compare the log average of the validation loss over the five cases to the same quantity for five direct trainings of identical model architectures using the protocol of the first experiment ($n_c = 2500$).

**SUPPLEMENTARY MATERIALS**

Supplementary material for this article is available at http://advances.sciencemag.org/cgi/content/full/7/26/eabf3658/DC1

**REFERENCES AND NOTES**

1. X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: LSTM cells and network architectures. Neural Comput. 31, 1235–1270 (2019).
2. I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks. Adv. Neural Inf. Process. Syst. 4, 3104–3112 (2014).
3. R. Pascanu, C. Gulcehre, K. Cho, Y. Bengio, How to construct deep recurrent neural networks, in Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014) (2014), pp. 1–13.
4. Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. 5, 157–166 (1994).
5. S. Hochreiter, J. Urgen Schmidhuber, Long short-term memory. Neural Comput. 9, 1735–1780 (1997).
6. K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014), pp. 1724–1734.
7. H. Salehinnejad, S. Sankar, J. Barfett, E. Colak, S. Valaee, Recent advances in recurrent neural networks. arXiv:1801.01078 (2017).
8. A. Graves, Generating sequences with recurrent neural networks. arXiv:1308.0850 (2013).
9. F. E. Bock, R. C. Aydin, C. J. Cyron, N. Huber, S. R. Kalidindi, B. Blussemann, A review of the application of machine learning and data mining approaches in continuum mechanics materials. Front. Mater. 6, 110 (2019).
10. L. Wu, V. D. Nguyen, N. G. Klingen, L. Noels, A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths. Comput. Methods Appl. Mech. Eng. 369, 113234 (2020).
11. M. Moazzar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, M. A. Bessa, Deep learning predicts path-dependent plasticity. Proc. Natl. Acad. Sci. U.S.A. 116, 26414–26420 (2019).
12. M. B. Gorji, M. Moazzar, J. N. Heidenreich, J. Cao, D. Mohr, On the potential of recurrent neural networks for modeling path dependent plasticity. J. Mech. Phys. Solids 143, 103972 (2020).
13. F. Ghamavami, A. Simone, Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. Comput. Methods Appl. Mech. Eng. 357, 112594 (2019).
14. A. Koeppe, F. Bamer, B. Markert, An efficient Monte Carlo strategy for elasto-plastic structures based on recurrent neural networks. Acta Mech. 230, 3279–3293 (2019).
15. A. L. Frankel, R. E. Jones, C. Allman, J. A. Templeton, Predicting the mechanical response of oligocrystals with deep learning. Comput. Mater. Sci. 169, 109099 (2019).
16. T. Young, D. Hazarika, S. Poria, E. Cambria, Recent trends in deep learning based natural language processing [Review Article]. IEEE Comput. Intell. Mag. 13, 55–75 (2018).
17. Abaqus, “1.1.7” in Abaqus 6.14 Documentation – Example Problems Guide Volume I (Dassault Systèmes, 2014).
18. Abaqus, “3.2.7” in Abaqus 6.14 Documentation – Benchmarks Guide (Dassault Systèmes, 2014).
19. Abaqus, “3.1.7” in Abaqus 6.14 Documentation – Example Problems Guide Volume II (Dassault Systèmes, 2014).
20. Abaqus, “2.1.13” in Abaqus 6.14 Documentation – Example Problems Guide Volume I (Dassault Systèmes, 2014).
21. C. Bonatti, D. Mohr, Data for “one for all: Universal material model based on minimal state-space neural networks”, Zenodo https://doi.org/10.5281/zenodo.4576777 (2021).
22. F. Chollet et al., Keras; https://keras.io.
23. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu,
Acknowledgments: We thank T. Tancogne-Dejean and J. Heidenreich for helpful discussions.

Funding: We acknowledge partial financial support through the Industrial Fracture Consortium.

Author contributions: C.B. and D.M. formulated the project, and D.M. supervised. C.B. conceived the idea of minimal state-space networks and performed the simulations and analysis. C.B. and D.M. discussed the results and co-wrote the manuscript. Competing interests: The authors declare that they have no competing interests.

Data and materials availability: All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. The codes and datasets used in this work are available at https://doi.org/10.5281/zenodo.4576777. Additional data related to this paper may be requested from the authors.

Submitted 21 October 2020
Accepted 11 May 2021
Published 23 June 2021
10.1126/sciadv.abf3658

Citation: C. Bonatti, D. Mohr, One for all: Universal material model based on minimal state-space neural networks. Sci. Adv. 7, eabf3658 (2021).
One for all: Universal material model based on minimal state-space neural networks
Colin Bonatti and Dirk Mohr

Sci Adv 7 (26), eabf3658,
DOI: 10.1126/sciadv.abf3658

ARTICLE TOOLS
http://advances.sciencemag.org/content/7/26/eabf3658

SUPPLEMENTARY MATERIALS
http://advances.sciencemag.org/content/suppl/2021/06/21/7.26.eabf3658.DC1

REFERENCES
This article cites 12 articles, 1 of which you can access for free
http://advances.sciencemag.org/content/7/26/eabf3658#BIBL

PERMISSIONS
http://www.sciencemag.org/help/reprints-and-permissions

Use of this article is subject to the Terms of Service