Acta Materialia 254 (2023) 118959

Contents lists available at ScienceDirect

Acta Materialia

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

# Robust deep learning framework for constitutive relations modeling

Qing-Jie Li [a,*], Mahmut Nedim Cinbiz [b], Yin Zhang [a], Qi He [a], Geoffrey Beausoleil II [b], Ju Li [a,c,*]

[a] *Department of Nuclear Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, United States*
[b] *Idaho National Laboratory, Idaho Falls, ID 83415, United States*
[c] *Department of Materials Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, United States*

A R T I C L E  I N F O

*Keywords:*
Constitutive-relation modeling
Artificial neural network
Machine learning
Materials mechanical behaviors

Modeling the full-range deformation behaviors of materials under complex loading and materials conditions is a significant challenge for constitutive relations (CRs) modeling. We propose a general encoder-decoder deep learning framework that can model high-dimensional stress-strain data and complex loading histories with robustness and universal capability. The framework employs an encoder to project high-dimensional input information (e.g., loading history, loading conditions, and materials information) to a lower-dimensional hidden space and a decoder to map the hidden representation to the stress of interest. We evaluated various encoder architectures, including gated recurrent unit (GRU), GRU with attention, temporal convolutional network (TCN), and the Transformer encoder, on two complex stress-strain datasets that were designed to include a wide range of complex loading histories and loading conditions. All architectures achieved excellent test results with an root-mean-square error (RMSE) below 1 MPa. Additionally, we analyzed the capability of the different architectures to make predictions on out-of-domain applications, with an uncertainty estimation based on deep ensembles. The proposed approach provides a robust alternative to empirical/semi-empirical models for CRs modeling, offering the potential for more accurate and efficient materials design and optimization.

## 1. Introduction

An mm-sized representative volume element (RVE) in an engineering continuum mechanics model can physically contain $\sim10^{20}$ atoms, $\sim10^{14}$ point defects, $\sim10^{5}$ m of mobile dislocations, numerous grain boundaries, hetero-phases, cracks, etc. It is clearly impossible to treat these degrees of freedoms (DOFs) on-the-fly for modeling and designing metal stamping, extrusion, crash-worthiness, etc., at the component scale. So the burden of coarse-graining over all these DOFs, which could certainly evolve under stress in a history-dependent manner, falls onto the so-called constitutive relations (CRs), as far as the macroscopic responses are concerned. The development of reliable CRs is critical for using computational mechanics to support component qualifications in automotive, aerospace and nuclear applications. Over the past many decades, various empirical/semi-empirical constitutive models have been developed. Based on the large strain behavior, plastic constitutive relations can be categorized as Voce type, Holomon type, or their combinations [1]. While Voce type relations tend to saturate [2–8], Holomon (power law) type of constitutive relations are unbounded at large strains [9,10]. These physically and phenomenologically based models have been widely used for different materials. However, one major limitation associated with these models is the insufficient capability to simultaneously describe various deformation stages under different conditions [11]. For example, most models only focus on specific stages of the entire deformation process and can only deal with a few external conditions (e.g., temperature and strain rate). Depending on the scenarios, many separate/modified models need to be constructed, such as in modeling various hardening behaviors [12–17]. For materials experiencing complex dynamic evolutions or materials optimization in a high-dimensional parameter space, a robust constitutive modeling calls for universal model (e.g., structural materials in nuclear applications can be simultaneously subjected to compositional/-structural/environmental changes).

Artificial neural networks (NN) have long been considered universal approximators [18,19], which naturally can serve as a robust alternative to empirical/semi-empirical constitutive modeling. The application of NN in constitutive modeling is not a new idea; some inspiring attempts were made nearly two to three decades ago. For example, Ghaboussi et al. [20] first introduced NN to model the mechanical behavior of plain concrete that was subjected to both biaxial loading and uniaxial cyclic loading. Ghaboussi et al. also proposed an auto-progressive algorithm for training NN to learn complex constitutive relations from global load-deflection response [21]. Lefik et al. implemented an incremental NN representation of constitutive relations that was successfully

* Corresponding authors.
*E-mail addresses:* qjl@mit.edu (Q.-J. Li), liju@mit.edu (J. Li).

https://doi.org/10.1016/j.actamat.2023.118959
Received 19 September 2022; Received in revised form 2 April 2023; Accepted 22 April 2023
Available online 23 April 2023

incorporated into a Finite Element (FE) code [22]. Hashash et al. [23] derived a consistent material stiffness matrix to address the numerical implementation issues for NN-based FE analysis. Jung and Ghaboussi [24] developed a rate-dependent NN visco-elasticity constitutive model and implemented it in FE analysis.

The dramatic boost in computational power and breakthroughs in newer NN architectures have led to a resurgence in constitutive modeling. Based on the inputs, three types of approaches can be identified for NN-based constitutive learning:

1 In the first approach, constitutive relations are learned from direct stress-strain responses. Such stress-strain data can be obtained from uniaxial or multi-axial loadings and include loading histories. For example, Gorji et al. developed a recurrent NN (RNN)-based framework for modeling the large deformation response of elasto-plastic solids subjected to arbitrary multi-axial loading paths [25].
2 Instead of learning constitutive relations from direct stress-strain data, the second approach learns the underlying constitutive relations from indirect measurements such as full field load-displacement data. For instance, Xu et al. [26] proposed an inverse modeling scheme to learn the constitutive relations of viscoelastic materials from indirect displacement data. Zhang et al. [27] developed a hybrid FE-NN framework to learn constitutive relations from full-field displacement data. Such indirect constitutive learning usually needs to consider physical constraints that are often in the form of partial differential equations.
3 The third approach takes advantage of prior knowledge, principles, or empirical models. For example, Li et al. [28] replaced the strain rate and temperature terms in Johnson-Cook model with a NN to describe the non-monotonic temperature dependence. Linka et al. [29] developed a CANNs (constitutive artificial neural networks) framework that learns a generalized strain energy function to link strain and materials information to stress responses. Incorporating prior knowledge or principles of mechanics and materials theory may reduce training data and achieve better extrapolation.

The mechanical response of materials is commonly characterized using simple, standardized uniaxial tensile test articles. The basic mechanical properties (elastic modulus, yield stress, ultimate stress, rupture strain, and ductility) and plastic constitutive relations (such as temperature sensitivity, strain and strain-rate hardening coefficients) are deduced from one-dimensional stress-strain curves. As the macroscopic strain-stress response is an effective and widely used measure of material deformation behaviors for engineering design, extensive strain-stress data sets are being accumulated from past and present research efforts. These data sets are also becoming more complicated as additional manufacturing methods yield additional features for consideration, e.g., through 3D printing and high throughput testing [30]. It is therefore beneficial to take advantage of strain-stress data to build robust constitutive models. In this work, we propose a general encoder-decoder deep learning framework to model complex high-dimensional stress-strain data. In the proposed framework, an encoder first projects the complex high-dimensional input data, such as loading histories, loading conditions, and materials information, onto a lower-dimensional hidden space. Such hidden representation of input information is then mapped to stresses of interest via a decoder. We evaluated a series of encoder architectures that are capable of modeling temporal data, such as gated recurrent unit (GRU) [31], GRU with attention, temporal convolutional network (TCN) [32], and the Transformer encoder [33]. The decoder was implemented as a fully connected network (FCN); however, it can be replaced with sequence modeling architectures if a series of stress prediction is needed. We tested all encoder architectures on two complex datasets: 1) a one-dimensional stress-strain curve with multiple unloading-reloading cycles to include complex loading history information; 2) a synthetic dataset based on the Johnson-Cook model [34] to include complex loading conditions such as temperature and strain rate. All architectures demonstrate excellent test results, with a root-mean-squared error (RMSE) well below 1 MPa, which is often needed to satisfactorily capture the full-range deformation features. We also explored the capability of different architectures on out-of-domain predictions with uncertainty estimation based on an ensemble of models [35]. Overall, our proposed general deep learning framework demonstrates high accuracy and universal capability for modeling various challenging CRs, thus offering a robust alternative to conventional CRs modeling for materials design and optimization.

## 2. Methods

### 2.1. Learning task and general encoder-decoder deep learning framework

From a fundamental point of view, the stress-strain relation of a material stems from the collective interatomic (and/or intermolecular) responses upon external loading, which involves various atomistic processes on different time-/length-scales. While the objective of our deep learning approach is to accurately and efficiently predict stress-strain relations in a physics-agnostic way, understanding the underlying deformation processes and material features on various spatial and temporal scales can be helpful in preparing meaningful training data. For example, to effectively model the macroscopic stress-strain relations, we may consider time-related data such as loading rate and temperature (as in thermally activated processes), and measurable material information on relevant lengthscale (e.g., chemical composition, crystal structure, defect population density). We can denote the measurable materials information using a vector **m** that may include composition, grain size, phase fractions, dislocation density, etc. Then, our learning task can be defined as

$$\boldsymbol{\sigma} = NN([\mathbf{m}, \boldsymbol{\varepsilon}, \mathbf{t}\ ]) \tag{1}$$

where *NN* means the NN model to be learned, **σ** is the stress, **m** is the measurable material information (note that **m** represents only the measurable part of the material microstructure, as much of the microstructural details are hidden from view and not measured), **ε** represents loading history, and **t** includes time-scale information such as strain rate, temperature, corrosion rate, radiation exposure, etc.

In the context of history-dependent NN modeling, the input information in Eqn. (1) is usually rewritten as a time sequence:

$$\mathbf{s} = \begin{bmatrix} \mathbf{x}_{t-n} \\ \cdots \\ \mathbf{x}_{t-2} \\ \mathbf{x}_{t-1} \\ \mathbf{x}_{t} \end{bmatrix} \tag{2}$$

where the component $\mathbf{x}_{t-i}$ in **s** is a vector containing input information at time $t-i$,

$$\mathbf{x}_{t-i} = \left[m^1_{t-i}, \ldots, m^{N1}_{t-i}, \varepsilon^1_{t-i}, \ldots, \varepsilon^6_{t-i}, \dot{\varepsilon}^1_{t-i}, \ldots, \dot{\varepsilon}^1_{t-i}, T_{t-i}\right] \tag{3}$$

Such input at time $t-i$ contains $N_1$ measurable materials information (although a subscript *t-i* is used, they are just constants serving to distinguish materials), 6 strain components, 6 strain-rate components, and temperature. Depending on specific scenarios, Eqn. (3) can be expanded/reduced to more/fewer components. For example, in the case of a single material (no need to distinguish from others) subjected to uniaxial tensile loading, different strain rates, and a range of temperatures, Eqn. (3) can be simplified as $\mathbf{x}_{t-i} = [\varepsilon^1_{t-i}, \dot{\varepsilon}^1_{t-i}, T_{t-i}]$ . Or if one wants to model a single uniaxial stress-strain relation (i.e., a single material at a fixed strain rate and temperature), then Eqn. (3) is further reduced to $\mathbf{x}_{t-i} = [\varepsilon^1_{t-i}]$. The output of a NN model (Eqn. (1)), in general, is a vector $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_6]$ that contains 6 stress components. However, **σ** can also be reduced to less components in cases such as uniaxial loading or two-

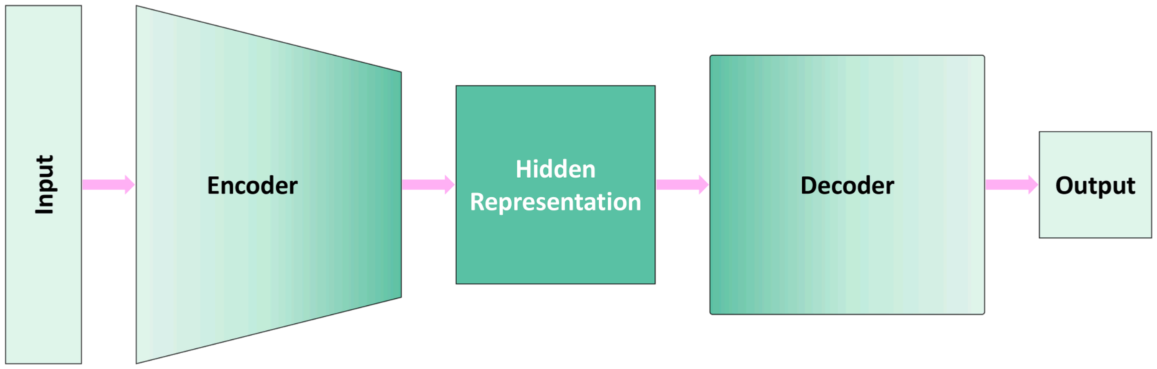

**Fig. 1.** The general encoder-decoder deep learning framework for modeling constitutive relations. The input is usually a high-dimensional sequence data that contain loading histories, loading conditions, and materials information. The encoder projects the high-dimensional input information onto a lower-dimensional hidden space, respecting the temporal correlations. The hidden representation is then mapped to stresses of interest via a decoder. The decoder can be a fully connected network or sequence modeling architectures depending on the need.

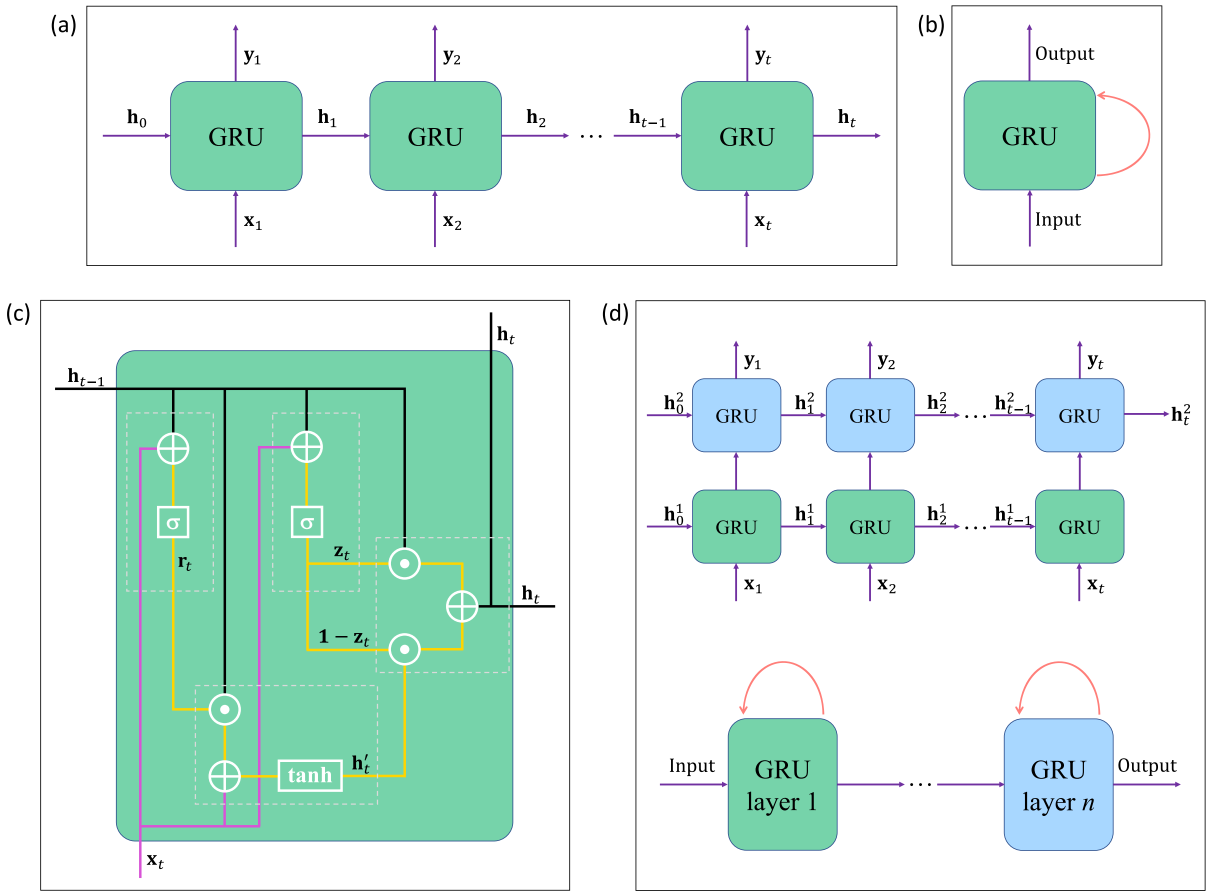

**Fig. 2.** Schematic illustration for GRU-based RNN architecture. (a) RNN consisting of GRU cells. The hidden state $\boldsymbol{h}_t$ at time t is updated based on the hidden state $\boldsymbol{h}_{t-1}$ and current input $\boldsymbol{x}_t$; an optional output $\boldsymbol{y}_t$ can also be generated. (b) A convenient representation for GRU-based RNN. (c) The gating mechanisms underlying a GRU cell. The information flow of hidden states and the input at time *t* are rendered in black and magenta, respectively, while the intermediate information flow is colored in yellow. Plus sign, sigma sign, circled dot, and tanh sign represents element-wise summation, sigmoid activation, Hadamard product, and tanh activation, respectively. Components associated with the reset gate $r_t$, update gate $z_t$, tentative new hidden states $\boldsymbol{h}_t^{'}$, and updated hidden state $\boldsymbol{h}_t$ are outlined with dashed lines rectangles, respectively. (d) An example of RNN consisting of two GRU layers (upper panel) and a convenient representation of multilayer GRU RNN.

dimensional loading.

To tackle the above outlined learning task, we propose a deep learning framework consisting of an encoder and a decoder (Fig. 1). The framework maps the high-dimensional input sequence data **s** to a lower-dimensional space and then learns to predict stresses of interest based on the encoded representation. The encoder plays a critical role in the framework by 1) capturing temporal correlations in the loading histories, and 2) effectively reducing the dimensionality of the input sequence data while preserving the most important information. Several popular architectures for handling sequence or temporal data, such as long short term memory (LSTM) [36], gated recurrent unit (GRU) [31], temporal convolutional network (TCN) [32], and Transformer [33], can

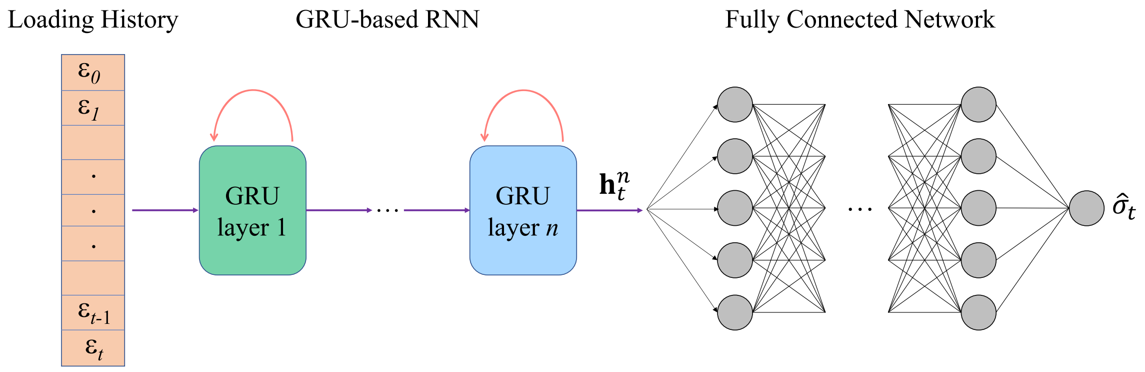

**Fig. 3.** GRU-encoder based architecture to model stress-strain responses.

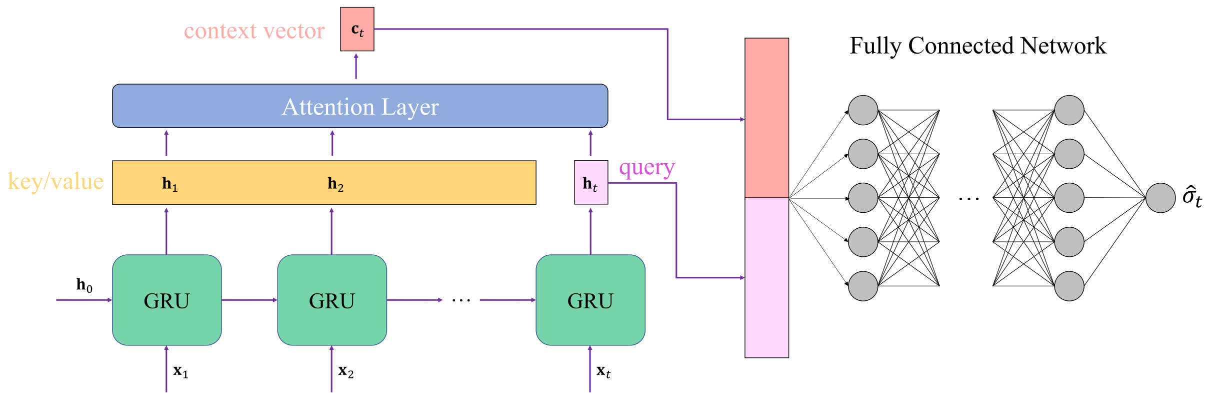

**Fig. 4.** GRU-attention based architecture for modeling stress-strain data. The attention layer uses the scaled dot-product attention mechanism, which takes the final hidden state from the last GRU layer as a query and previous hidden states (after separate linear transformations) as keys and values, and outputs a context vector. The context vector is concatenated with the final hidden state to form a hidden representation which is then fed into the FCN decoder to predict stress of interest.

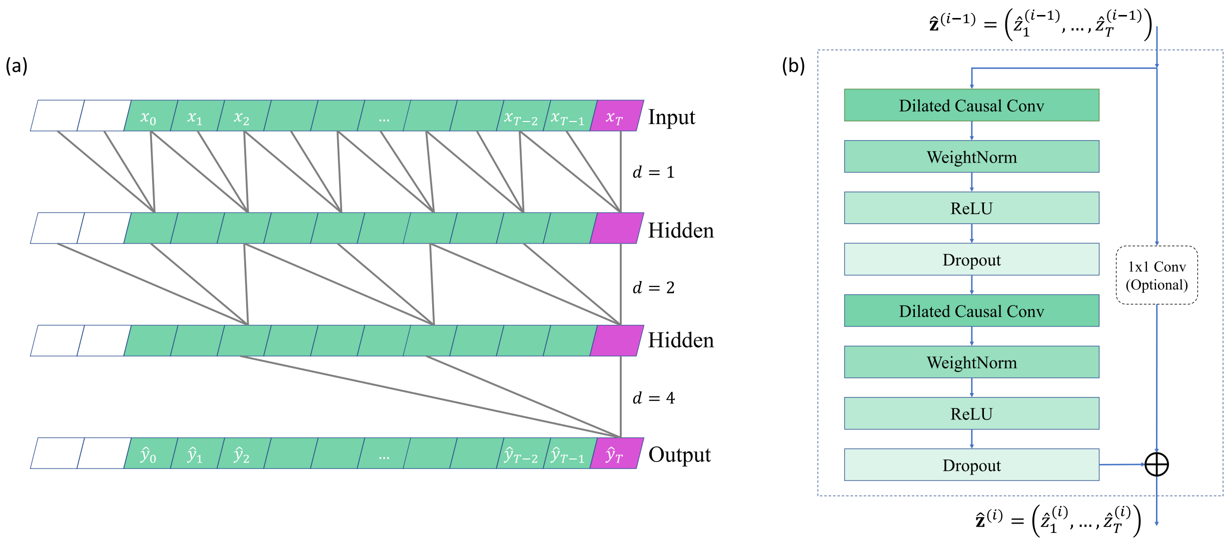
(a)
$x_0$ $x_1$ $x_2$ ... $x_{T-2}$ $x_{T-1}$ $x_T$
Input
$d = 1$
Hidden
$d = 2$
Hidden
$d = 4$
$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
(b)
$\hat{\mathbf{z}}^{(i-1)} = \left(\hat{z}_1^{(i-1)}, \ldots, \hat{z}_T^{(i-1)}\right)$
Dilated Causal Conv
WeightNorm
ReLU
Dropout
Dilated Causal Conv
WeightNorm
ReLU
Dropout
1x1 Conv (Optional)
$\hat{\mathbf{z}}^{(i)} = \left(\hat{z}_1^{(i)}, \ldots, \hat{z}_T^{(i)}\right)$

**Fig. 5.** Architecture of temporal convolutional network. (a) Causal and dilated convolutions in a 1D convolutional network. Zero paddings (white cells) are applied to ensure the same length for each convolutional layer. (b) Residual block as employed in the TCN architecture. The residual block consists of two dilated causal convolutional layers, each of them followed by weight normalization, ReLU activation, and spatial dropout operations. A skip connection is made from the input to the output of the second Dropout operation. A $1 \times 1$ convolution will be used if the residual input and output have different dimensions. More details can be found in Ref. [32].

be used as the encoder. These architectures have already achieved impressive results in natural language processing and time series forecasting. By explicitly imposing the temporal constraint in these encoder architectures, the model training will be guided to more physically relevant regions in the parameter space, thus potentially improving its extrapolation capability. For the decoder, we can use a fully connected

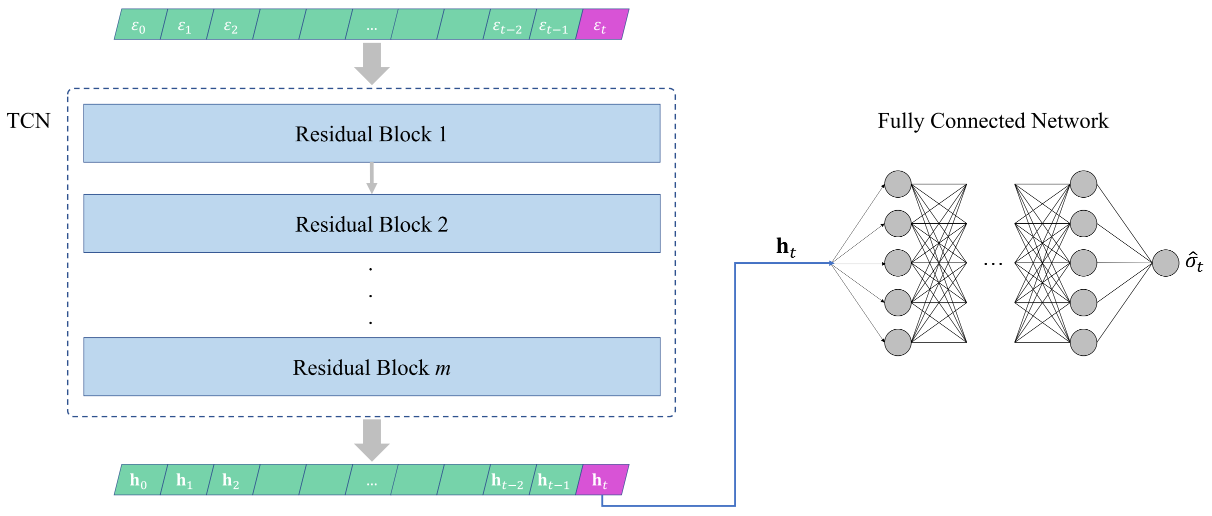

**Fig. 6.** Temporal convolutional network based architecture to model strain-stress relation. The TCN takes a loading history consisting of a strain sequence as input and outputs the hidden states at each time. The hidden state at time $t$, $\mathbf{h}_t$, is then passed to a FCN decoder to predict the stress at time $t$, $\hat{\sigma}_t$. Multiple channels/filters can be used in each of the residual block of TCN, thus the hidden state $\mathbf{h}_t$ is a vector in general.

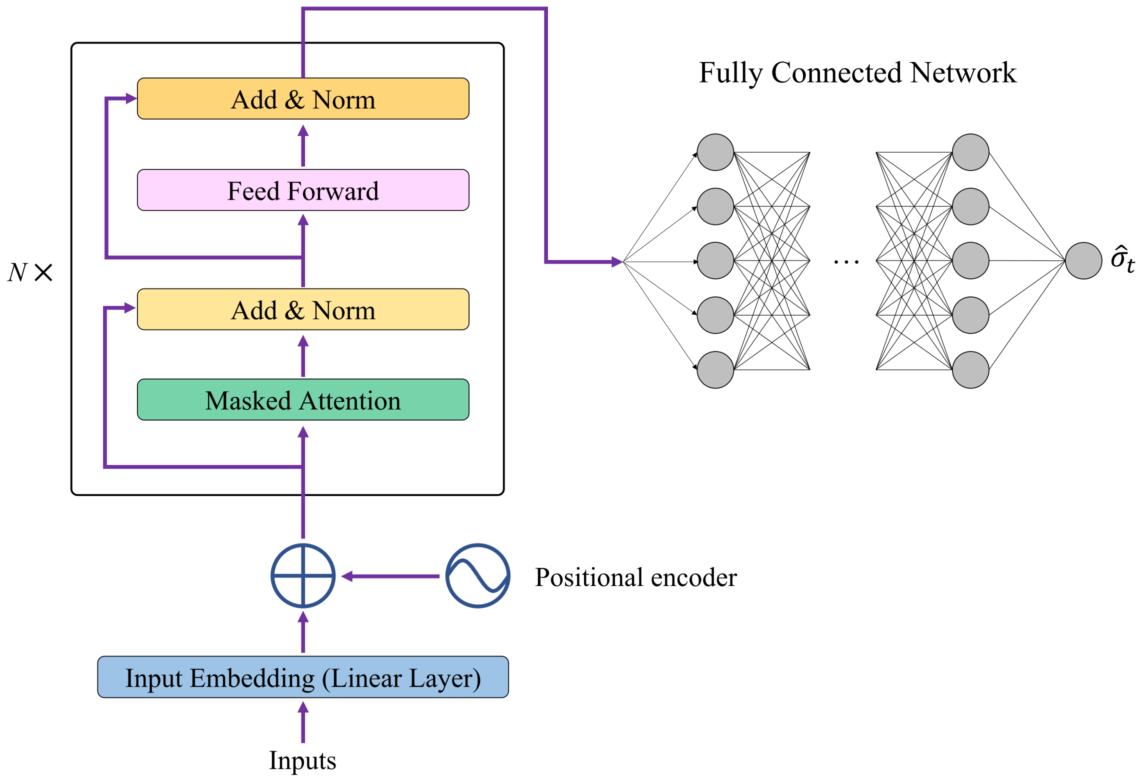

**Fig. 7.** Transformer-encoder based architecture to model strain-stress relation. The transformer-encoder replaces the recurrent neural network for modeling sequence data by incorporating positional information through a positional encoder and using a masked attention mechanism to capture important history information. The output of the transformer encoder is a sequence, from which only the last component is utilized as the hidden representation of input information. The hidden representation is then mapped to stresses via a fully connected network.

network (FCN) if the output is a single scaler stress, or relevant sequence modeling architectures if the output is a series of stresses. The availability of various encoder/decoder architectures provides a rich set of tools to choose from and enables the framework to adapt to different types of data and modeling objectives. In the following, we briefly introduce several encoder architectures relevant to this work.

### 2.2. *Gated recurrent unit-based framework*

Gated recurrent unit (GRU) [31], a variation of the LSTM network, is a type of recurrent neural network (RNN) that can effectively retain long-term dependencies in sequential data such as speech and text, and has been widely adopted for natural language processing tasks, as well as sensing and guiding dynamical actions with policy NN in gaming and robotics. Here we use GRU to process loading history in an RVE. As

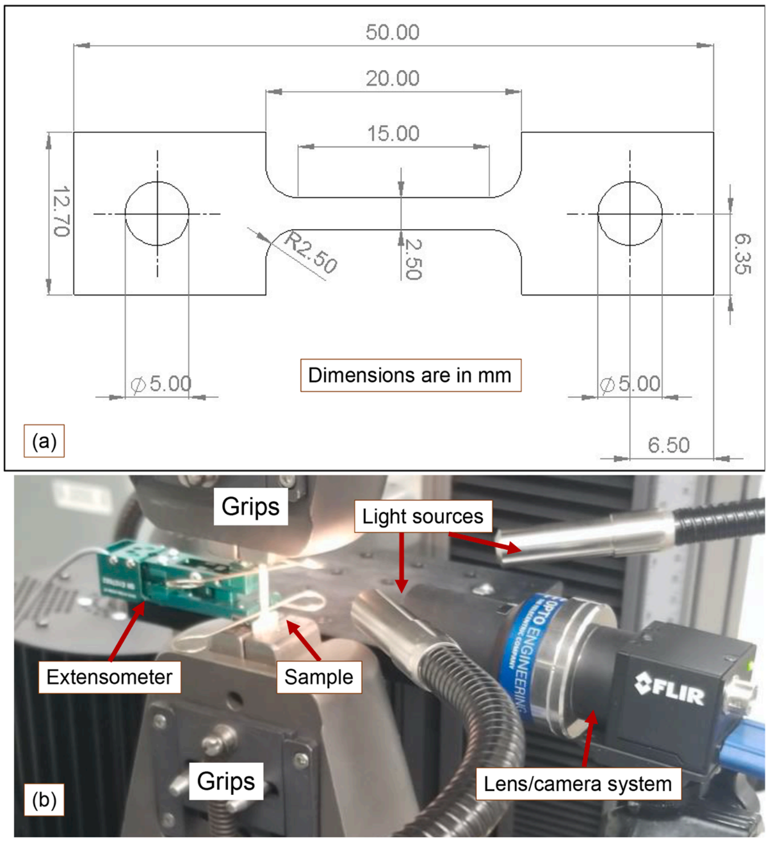

**Fig. 8.** Experimental tensile tests (a) specimen geometry and (b) experimental setup.

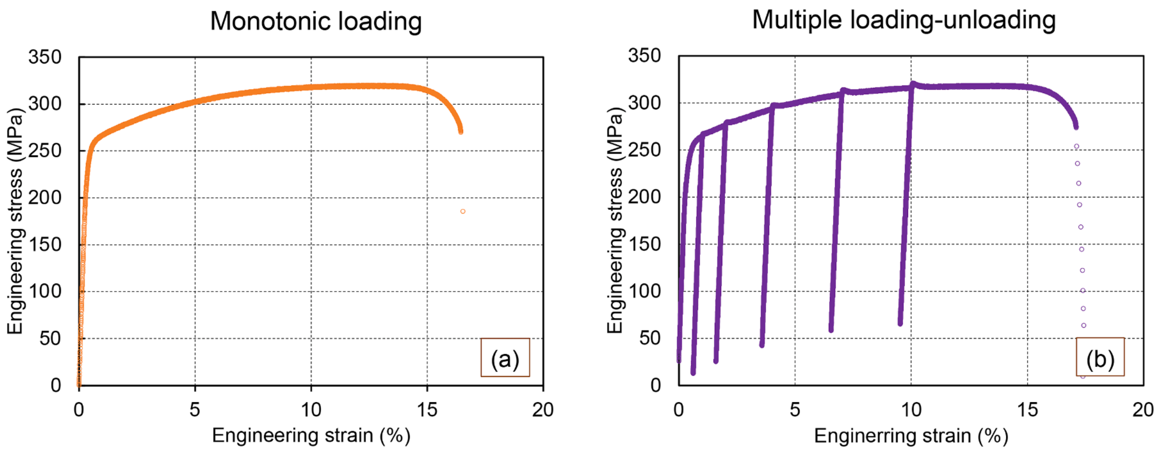

**Fig. 9.** Engineering stress and strain curves of (a) monotonically increasing and (b) multiple loading-unloading test specimens.

**Table 1**
Unloading Basic mechanical properties of the aluminum sheet samples derived from experimental mechanical tests.

| Test type | Yield stress (MPa) | Ultimate tensile stress (MPa) | Total elongation (%) |
|---|---|---|---|
| Monotonically increasing load | 253.7 | 319.3 | 16.5 |
| Multiple loading-unloading | 251.0[a] | 320.0 | 17.4 |

[a] based on the first loading portion.

**Table 2**
Plastic strains and stresses at interruptions.

| Plastic strain (-) | Stress (MPa) |
|---|---|
| 0.006 | 263 |
| 0.0148 | 275 |
| 0.0345 | 289 |
| 0.0642 | 307 |
| 0.094 | 315 |

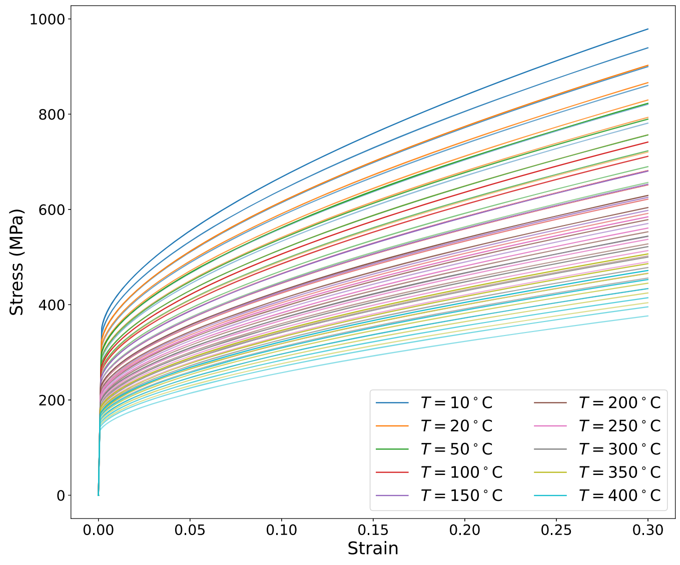

**Fig. 10.** Stress-strain relations from the second datasets. These stress-strain curves were generated based on the Johnson-Cook model for AISI 316 L steel [34]. Each sampled temperature also cover the stress-strain relations at six strain rates, $10^{-4}$ s$^{-1}$, $10^{-2}$ s$^{-1}$, $10^{0}$ s$^{-1}$, $10^{2}$ s$^{-1}$, $10^{4}$ s$^{-1}$, and $10^{6}$ s$^{-1}$.

shown in Fig. 2(a), in a GRU-based RNN, a GRU cell takes 1) the hidden state $\mathbf{h}_{t-1}$ from previous time $t$-1, and 2) current input $\mathbf{x}_t$ as inputs to update the current hidden state

$$\mathbf{h}_t = f(\mathbf{h}_{t-1},\ \mathbf{x}_t) \tag{4}$$

where $f$ represents the GRU nonlinear activation function. A more compact representation of the GRU-based RNN is shown in Fig. 2(b). GRU uses gating mechanisms to control information flow between recurrent cells. As shown in Fig. 2(c), to compute the hidden state $\mathbf{h}_t$, we first compute a *reset gate* $\mathbf{r}_t$ and an *update gate* $\mathbf{z}_t$, separately, according to

$$\mathbf{r}_t = \text{sigma}(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r) \tag{5}$$

$$\mathbf{z}_t = \text{sigma}(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z) \tag{6}$$

where $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are weight/bias matrices/vectors to be learned (subscripts r and z correspond to reset and update gate, respectively), and sigma$(\cdot)$ is the sigmoid activation function. Then the reset gate $\mathbf{r}_t$ is further used to compute a *candidate hidden state* $\widetilde{\mathbf{h}}_t$

$$\widetilde{\mathbf{h}}_t = \tanh(\widetilde{\mathbf{W}}\mathbf{x}_t + \widetilde{\mathbf{b}}_x + \mathbf{r}_t \odot (\widetilde{\mathbf{U}}\mathbf{h}_{t-1} + \widetilde{\mathbf{b}}_h)) \tag{7}$$

where $\widetilde{\mathbf{W}}$, $\widetilde{\mathbf{U}}$, and $\widetilde{\mathbf{b}}$ are learnable weight/bias matrices/vector and tanh is the hyperbolic tangent activation function. Finally, the hidden state at the current time $t$ is updated as

$$\mathbf{h}_t = \mathbf{z}_t\mathbf{h}_{t-1} + (1 - \mathbf{z}_t)\widetilde{\mathbf{h}}_t \tag{8}$$

If the reset gate $\mathbf{r}_t \sim 0$, the information from previous hidden states will be dropped out and $\widetilde{\mathbf{h}}_t$ will be reset with current input information, while the update gate $\mathbf{z}_t$ determines how much information from the previous hidden state should be retained. Together, such gating mechanisms can capture the most relevant information from a long sequence of data stream for predicting the quantity of interest. Operations associated with Eqn. (5)-(8) are outlined in Fig. 2(c) using dashed rectangles. One may even stack multiple GRU-based RNNs to form a multilayer architecture, as shown in Fig. 2(d). In this case, hidden states for the first GRU layer are updated using the same procedure as above. For an intermediate layer $i$, the hidden state at time $t$, $\mathbf{h}_t^i$, is updated according to

$$\mathbf{h}_t^i = f\left(\mathbf{h}_{t-1}^i, \mathbf{h}_t^{i-1}\right) \tag{9}$$

where $\mathbf{h}_{t-1}^i$ is the previous hidden states in layer $i$ and $\mathbf{h}_t^{i-1}$ is the current hidden state in previous layer $i$-1. Such multilayer GRU-based RNN allows further abstraction of the raw sequence data.

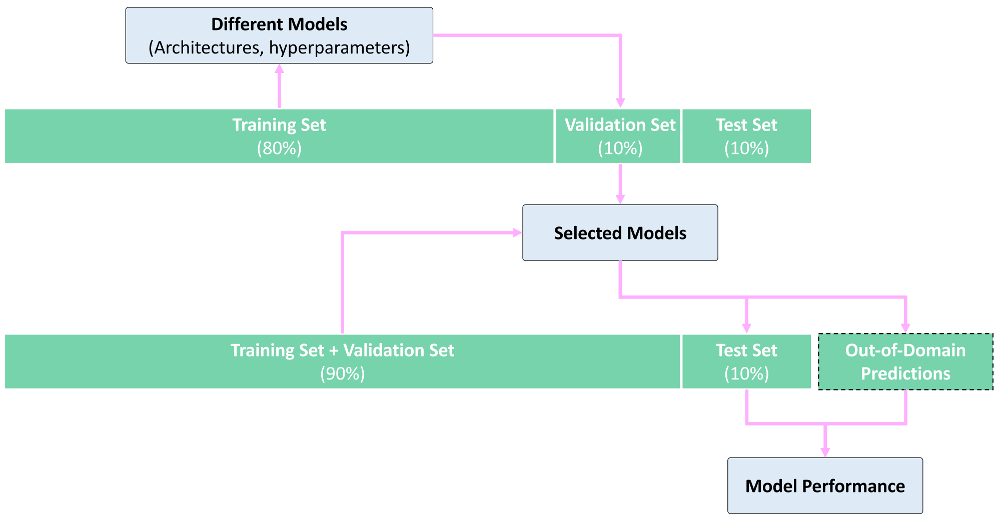

**Fig. 11.** Schematic illustration of dataset splitting and model selection/testing. The entire dataset is split into training/validation/test sets at a ratio of 8:1:1. The training set is used to train a wide range of models using different architectures and different hyperparameters. These initially train models were then validated using the validation set, to select the optimal model hyperparameters. Selected models were retrained on the combined training and validation set. Finally, the test set was used to obtain the model performance. The selected models can also be tested on out-of-domain tasks to evaluate their predictive capabilities.

**Table 3**
Architecture hyperparameters considered in model selection.

| | GRU | | GRU-Attention | | | TCN | Transformer | |
|---|---|---|---|---|---|---|---|---|
| | hidden size | GRU layers | hidden size | GRU layers | query/value size | channel numbers | hidden size | encoder layers |
| set-1 | 5 | 3 | 5 | 3 | 5/5 | 3 | 5 | 2 |
| set-2 | 5 | 6 | 5 | 6 | 5/5 | 5 | 5 | 4 |
| set-3 | 5 | 9 | 5 | 9 | 5/5 | 8 | 10 | 2 |
| set-4 | 10 | 6 | 10 | 10 | 5/10 | 10 | 12 | 2 |

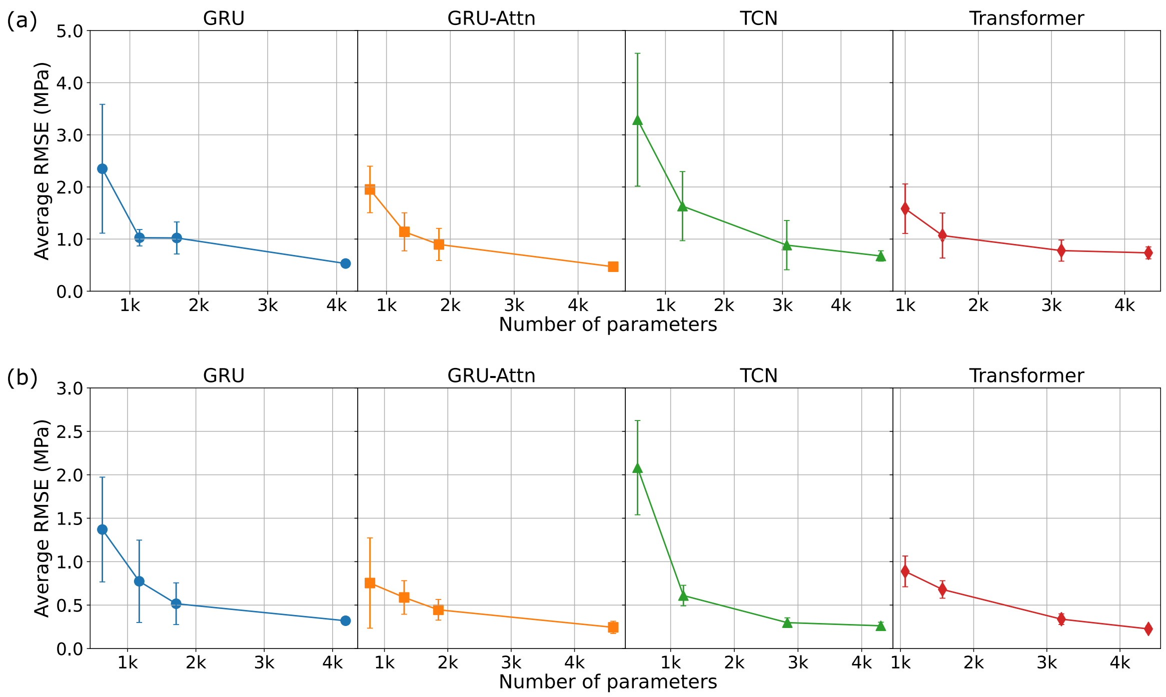

**Fig. 12.** Validation results on different models. (a) Dataset 1 models. (b) Dataset 2 models.

Let us take uniaxial loading as an example to demonstrate how to fit GRU into the encoder-decoder framework. To predict the current stress $\sigma_t$ for a given loading history vector $\boldsymbol{\varepsilon} = \{\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{t-1}, \varepsilon_t\}$, we first extract the final hidden state vector $\mathbf{h}_t^n$ from a $n$-layer GRU encoder. Then $\mathbf{h}_t^n$ is passed to an FCN that further outputs the predicted stress $\widehat{\sigma}_t$. The entire architecture is schematically shown in Fig. 3. Gorji et al. [25]. successfully applied a similar architecture to predict the stress-strain responses for materials.

### 2.3. GRU-attention based framework

The GRU-encoder architecture described above only uses the final hidden states $\mathbf{h}_t^n$ from the last GRU layer, and discards all the previous hidden states in the same layer. However, all these previous hidden states contain valuable historical information and are easily accessible from computer memory. By properly retrieving information from these stored hidden states, we can obtain useful contextual information. One effective way to achieve this is by using an attention mechanism [37]. This mechanism allows the decoder to automatically focus on the most relevant parts of the previous hidden states. The scaled dot-product attention mechanism [33] has been shown effective in different tasks:

$$\text{Attention}(Q,\ K,\ V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{10}$$

where $Q$ is a query, $K$ is the keys interacting with the query, $d_k$ is the dimension of the query vector and key vectors, $V$ is the values to be summed with a weight factor from the softmax function. In our case, the final hidden state $\mathbf{h}_t^n$ serves as a query $Q$ which interacts (via dot-product) with all previous hidden states (keys $K$, which are obtained through a linear transformation of previous hidden states), to determine how much information (softmax function) from each previous hidden state should be retrieved (values $V$, which are obtained via another linear transformation of previous hidden states). Such attention layer outputs a context vector $c_t$, which is then concatenated with $\mathbf{h}_t^n$ to form the hidden representation of input sequence data. This hidden representation is then mapped to stresses through a FCN decoder. The overall architecture is shown in Fig. 4.

### 2.4. Temporal convolutional network (TCN) based framework

Temporal convolutional network (TCN) [32] has been shown as a robust method for various sequence modeling tasks [32]. The original TCN architecture tries to predict an output sequence $y_0, \ldots, y_T$, given an input sequence $x_0, \ldots, x_T$:

$$\widehat{y}_0,\ \ldots,\ \widehat{y}_T = f(x_0, \ldots, x_T) \tag{11}$$

Such mapping must satisfy a causal constraint that each of the component $y_t$in the output sequence only depends on $x_0, \ldots, x_t$, meaning that there is no information "leakage" from the future to the past. Usually the causality constraint is achieved using *causal convolutions* in a 1D fully convolutional network, Fig. 5(a), i.e., the output at time $t$ is convolved only with elements up to $t$ in the previous layer. For example, in Fig. 5(a), the last element (magenta) in the first hidden layer is convolved only with $x_{T-2}, x_{T-1},\ x_T$ if the filter size $k = 3$. However, such simple causal convolution may require very deep network to span a sufficiently long history (i.e., the receptive field). To solve this problem, dilated convolutions are usually employed to effectively increase the receptive field. For example, if a dilation factor $d = 2$ is used, as shown in Fig. 5(a) from the first hidden layer to the second hidden layer, a fixed step of 2 will be introduced between two adjacent filter taps. A common practice is to exponentially increase the dilation factor $d$ with the depth of the network. Overall, the receptive field of TCN depends on the network depth, filter size $k$, and dilation factor $d$. For very large history size (e.g., $10^3$), even if we employ dilation convolution with a relatively large filter size $k$, it may still require a very deep network, which can lead to serious stabilization issues of TCN. Therefore, the convolutional layer, as shown in Fig. 5(a), is often replaced by a residual block (Fig. 5(b)) that learns the residual to the identity mapping of input, which benefits deep network learning tasks. In Fig. 5(b), the residual block consists of two dilated causal convolutional layers, each of them followed by weight normalization, the ReLU nonlinear activation, and spatial dropout for regularization. See Ref. [32] for more details. We note that Ghaboussi and Sidarta [38] in 1998 proposed a nested adaptive neural network (NANN) architecture to account for loading history dependence. In NANN, more distant loading history only has a one-way connection to more recent loading history, which also prevents

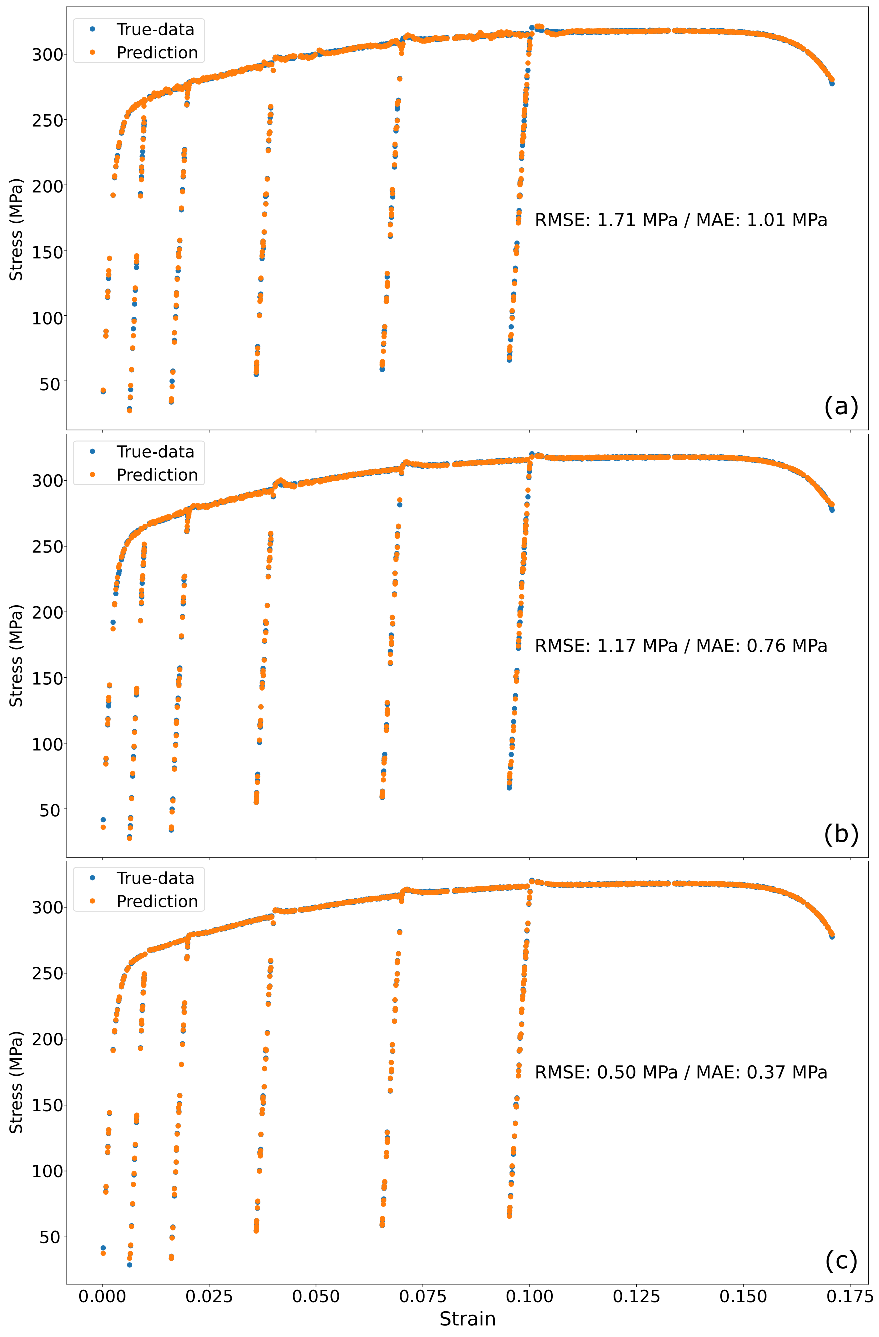

**Fig. 13.** The effects of model accuracy on predicted stress-strain curves. (a-c) three GRU models showing relatively large, medium, and small RMSE on the validation set. When the validation RMSE converges below 1 MPa, all features, such as elastic deformation, strain hardening, and sharp transitions during unloading-reloading, etc., are correctly captured.

information leakage from the future.

Fig. 6 shows the overall encoder-decoder architecture using TCN as an encoder. As seen in the case of a uniaxial loading, the TCN encoder takes a loading history vector $\boldsymbol{\varepsilon} = (\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_t)$ as input and outputs the hidden states associated with each input element, $\mathbf{h} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_t)$. Then the hidden state $\mathbf{h}_t$ corresponding to $\varepsilon_t$ is chosen as hidden representation of the input sequence which is passed to an FCN decoder to predict stress $\widehat{\sigma}_t$. Note that one can use multiple channels/filters for the causal dilated convolutional layers in TCN, thus $\mathbf{h}_t$in general is a vector that contains the most relevant loading history information to predict the current stress.

## 2.5. *Transfomer encoder based framework*

The Transformer architecture [33] has become a key component of natural language processing and is widely used in applications such as machine translation, sentiment analysis, and text generation. Unlike recurrent neural networks that process sequences sequentially, the Transformer architecture uses a novel self-attention mechanism that allows parallel processing of sequence. This achieves faster training and inference, as well as better performance on long-range dependences in sequence modeling tasks. In this work, we utilize the Transformer encoder to learn the hidden representation of input information.

As illustrated in Fig. 7, our framework first projects all components in the input sequences to a higher-dimensional space using a linear layer.

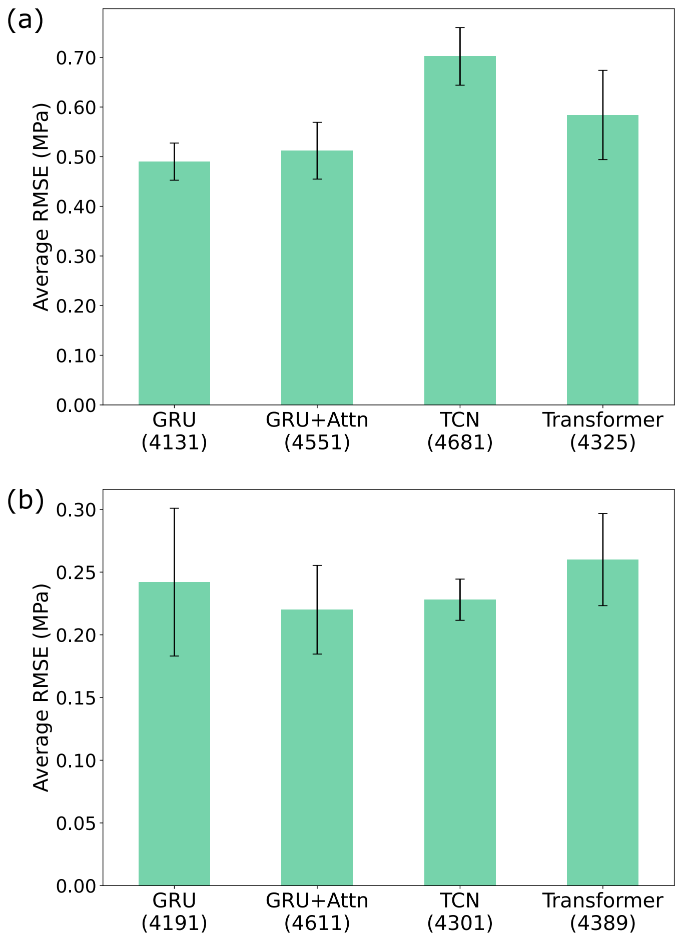

**Fig. 14.** Testing results for different types of encoders using selected model hyperparameters. (a) Testing results on the test set of dataset 1. (b) Testing results on the test set of dataset 2. The average RMSE is based on five different models and the error bar represents the standard deviation of RMSEs. The number in parenthesis means the number of model parameters.

Then, a positional encoder adds positional information to the linearly transformed input. The encoded input sequence is then fed into the Transformer encoder layer, which consists of a masked attention layer and a feed forward network. The masked attention layer only allows a query to attend to all previous components, thereby preventing information leakage from future components. To stabilize gradients and speeds up convergence during training, an "Add & Norm" layer was added after the attention layer and feed-forward network to perform residual connections and layer normalization. Multiple encoder layers can be stacked to carry out iterative attention operations. The encoder ultimately outputs a sequence that contains the most relevant historical information at each component.

To obtain the hidden representation of the input sequence, we use only the last component from this output sequence, as it contains the most relevant historical information from the entire input sequence. Finally, we map the hidden representation to stresses via a fully connected network. Overall, our framework utilizes the Transformer encoder to effectively learn the hidden representation of input information while leveraging the benefits of the self-attention mechanism.

### 2.6. *Datasets and model training/validation/testing*

We generate two datasets to test the capability of various encoder-decoder architectures. The first dataset was obtained from uniaxial tensile tests of aluminum sheet samples. Tensile specimen design was based on previous uniaxial tensile specimen design with a gage length-to-width ratio greater than 4 [39–41]. The specimen design and experimental setup are shown in Fig. 8.

The test procedure followed the American Society for Testing and Materials (ASTM) E8/8M-21 standard [42] for tensile testing of metallic materials. While optical metrology data was available, the strain data was collected using a pre-calibrated Epsilon extensometer (3442–015M-050M-ST SN#E107682), and the load was measured using a calibrated 5 kN Instron load cell. Both monotonically increasing and multiple loading-unloading tests, as shown in Fig. 9, were performed for the constitutive relation determination. Basic mechanical properties are listed in Table 1.

By applying the power law constitutive relationship to the multiple loading-unloading data (see Table 1), the strain hardening coefficient ($n$) and the constant ($K$) were calculated as 0.066 and 366 MPa, respectively. The interruption strains for unloading are shown in Table 2.

The use of multiple unloading/reloading paths also made the learning task more challenging. The entire dataset consists of ~8500 stress-strain pairs with a total strain up to ~18%. The loading history vector length should be chosen such that all important past points should be included to inform current decision. For example, if we want to predict the stress during an unloading process, then we should at least include the previous load-unload transition region and some strain hardening region before the load-unload transition. We used a loading history vector that contains 600 strains for each training sample (zero padding is properly added for the first zero strain).

The second dataset was generated from the Johnson-Cook model for AISI 316 L steel [34], covering a wide range of loading conditions. Specifically, we included the stress-strain relations in a wide temperature range, from 10°C to 400°C. For each sampled temperature, we also included the stress-strain relations at six different strain rates, i.e., $10^{-4}$ s$^{-1}$, $10^{-2}$ s$^{-1}$, $10^{0}$ s$^{-1}$, $10^{2}$ s$^{-1}$, $10^{4}$ s$^{-1}$, and $10^{6}$ s$^{-1}$. See Fig. 10 for a plot of these stress-strain relations. Each stress-strain curve has 3230 stress-strain pairs up to a total strain of 30%. We use a loading history vector of length 500 and pad zero values for the very first zero strain (such that it has the same history length). Thus, each stress-strain curve provides 3230 sequence data, each of which contains its own loading history, temperature, and strain rate information.

Both datasets were split randomly into three subsets: a training set, a validation set, and a test set, using an 8:1:1 ratio. The training set was used to train different models, while the validation set was used to select optimal model hyperparameters. After obtaining the optimal model hyperparameters, each selected model was retrained on a combined dataset consisting of the training and validation sets. Finally, the test set was used to evaluate the performance of the models. The data splitting, model training/validation/testing process is shown in Fig. 11.

To ensure that numerical scales did not influence the model training process, we performed data normalization. While strains were kept on their original scale, the temperature, strain rate, and stress quantities were standardized to have a mean of zero and a standard deviation of 1. The strain rates were log-transformed before the standardization. It's important to note that all standard scalers were obtained based on the training set and then applied to the validation and test tests, as well as any unseen data.

The PyTorch [43] package was used for all architecture implementation and model training. Dataset 1 models were trained for 1000 epochs, while dataset 2 models were trained for 200 epochs. An initial learning rate of $10^{-3}$ was used for all training processes, which then decays every epoch with a multiplicative factor of 0.996. We used the Adam optimizer with default hyperparameters. Model training utilized a batch size of 100 and model performance was evaluated using the root-mean-squared error (RMSE).

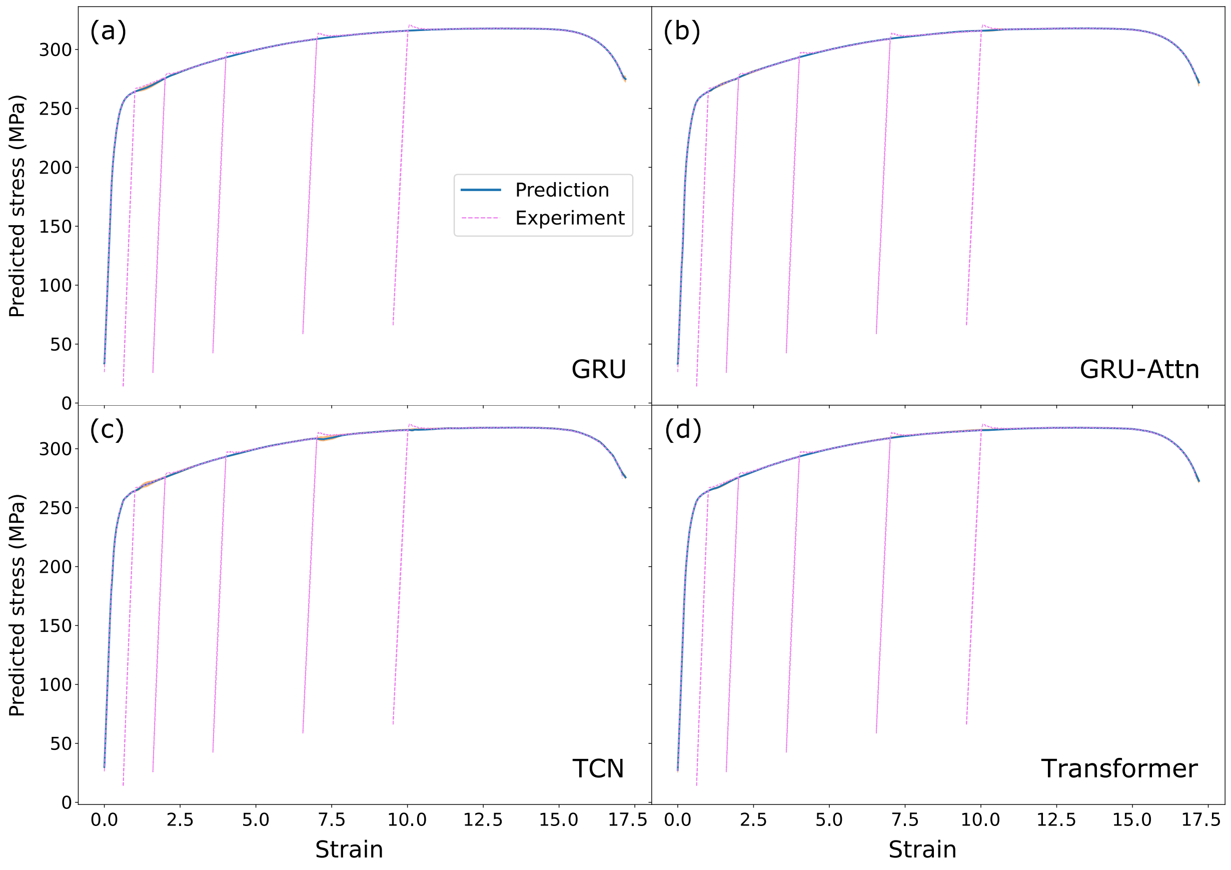

**Fig. 15.** Predictions on uniaxial loading without any unloading/reloading cycles. (a) Prediction from GRU-based models. (b) Predictions from GRU-attention-based models. (c) Predictions from TCN-based models. (d) Predictions from the Transformer encoder-based models. Such monotonic uniaxial loading path was not seen during model training. For each prediction, the blue solid line is the average prediction from five different models and the shaded band indicates the uncertainty (based on the standard deviation of the predictions). Strains are in percentage.

## 3. Results and discussions

### 3.1. *Model validation and testing*

To determine the optimal set of hyperparameters for each architecture, we performed model validation by training various models on the training set and evaluating their performance on the validation set. For each architecture, we started with a simple model and gradually increased its complexity to identify a robust model with a small number of model parameters as possible. Table 3 lists the important hyperparameter choices for each architecture. For the GRU encoder, we considered two critical hyperparameters: the size of the hidden state vector of the GRU cell and the number of GRU layers. In addition to these hyperparameters, for the GRU-attention encoder, we also took into account the query/value sizes, which dictate the sizes of keys/values in the attention layer. For the TCN encoder, we set the kernel size and dialation base to 2, and only focused on varying the number of channels for the causal convolution. Lastly, for the Transformer encoder, we examined the hidden vector size resulting from the attention operation and the number of encoder layers. The input embedding size and the positional encoder size were set to 32. In all cases, we used a uniform FCN decoder consisting of two hidden layers. The first hidden layer was twice the size of the hidden representation, while the second hidden layer was of the same size as the hidden representation. The output layer contained only one unit.

Fig. 12 illustrates the RMSE on the validation sets of dataset 1 and dataset 2. We varied the number of model parameters (weights and biases) by exploring different hyperparameter sets (set-1 to set-4 in Table 3). For each hyperparameter set, we trained 10 different models with different parameter initializations. The markers in Fig. 12 indicate the average RMSE over these models, and the error bars represent the standard deviations of these RMSEs. As expected, the average RMSE decreases as the model size increases. However, there are two ways to increase model size: 1) making the network deeper, as in the case of GRU and GRU-attention encoders, where the first three models (set-1 to set-3 in Table 3) have the same hidden representation size (5) but different numbers of GRU layers; and 2) using larger sizes for hidden representations, as in set-2 and set-4 for both GRU and GRU-attention encoders, set-1 to set-4 for TCN encoders, and set-1, set-3, set-4 for the Transformer encoder. Generally, deeper networks with proper connections lead to better performance, while larger hidden representations could provide more representation capability. However, increasing model complexity should be done cautiously to avoid overfitting when the training data is limited. We found that a validation RMSE smaller than 1 MPa can satisfactorily capture various stress-strain relations, including elastic deformation, strain hardening, and sharp transition during unloading-reloading cycles. Fig. 13 shows examples of models with different validation errors: the model with the largest validation error shows relatively poor prediction on strain hardening and elastic deformation, the model with medium validation error predicts strain hardening well but falls short in elastic deformation prediction, and the model with a validation error below 1 MPa captures all features in the full range deformation. Therefore, we stopped increasing the model size once the validation error converged below 1 MPa. Based on these validation results, we selected set-4 hyperparameters for each type of encoder for the testing and out-of-domain predictions.

We then performed model testing using the selected set-4 hyperparameters by retraining five different models for each type of encoder

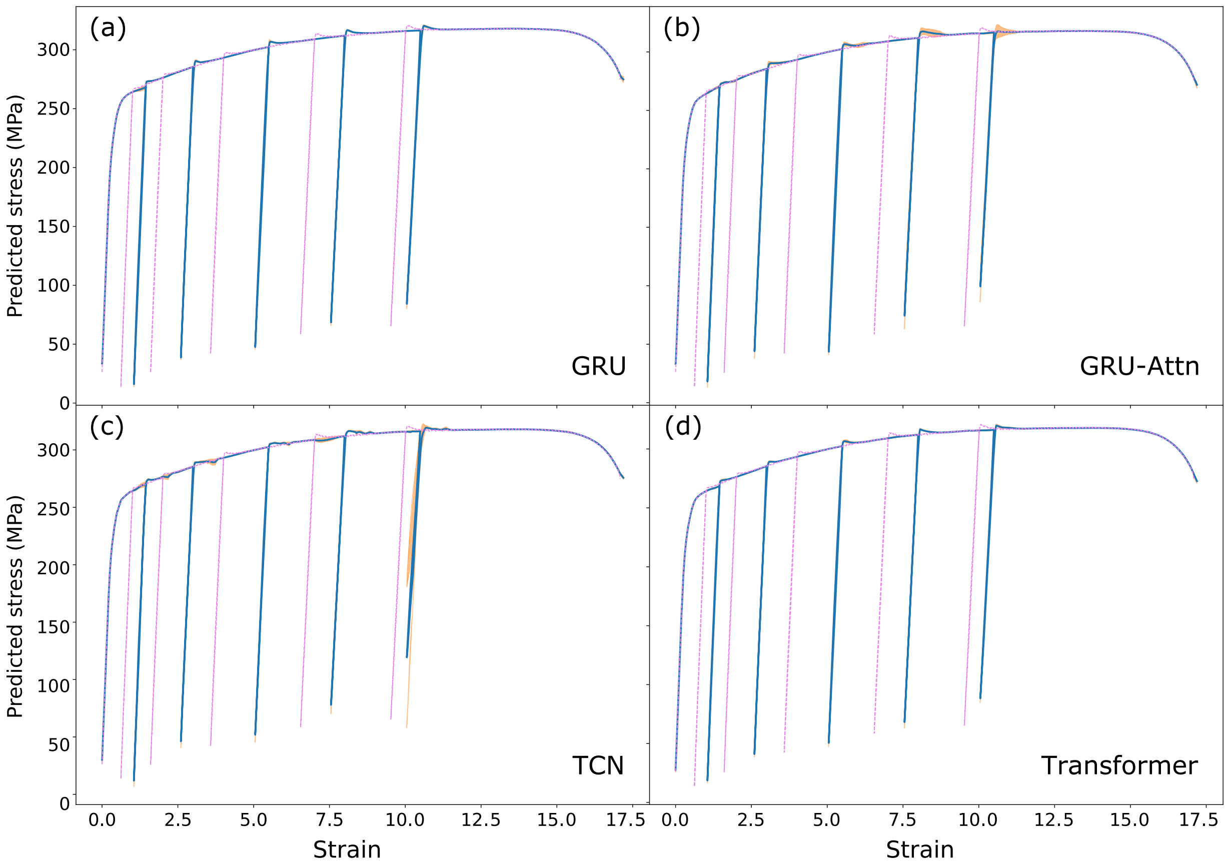

**Fig. 16.** Predictions on unseen unloading/reloading cycles. (a) Prediction from GRU-based models. (b) Predictions from GRU-attention-based models. (c) Predictions from TCN-based models. (d) Predictions from the Transformer encoder-based models. For each prediction, the blue solid line is the average prediction from five different models and the shaded band indicates the uncertainty (based on the standard deviation of the predictions). Dashed lines are experimental stress-strain relations. Strains are in percentage.

with different parameter initializations. We retrained each model on a combined dataset consisting of the original training set and validation set (see Fig. 11 for the illustration), while the original test sets were used to test these retrained models. As shown in Fig. 14, these models generally have less than 5000 parameters, but the average testing RMSE was below 0.7 MPa and 0.3 MPa for dataset 1 and dataset 2 tasks, respectively. Such testing results are comparable to the validation RMSEs shown in Fig. 12, thus we believe there is no obvious overfitting. For the dataset 1 task, we can see that GRU-based model showed the smallest testing RMSE, while GRU-attention-based model showed similar performance. The TCN-based model had the largest testing RMSE even though it had the largest model sizes, and the Transformer-based model showed intermediate testing RMSE. For the dataset 2 task, GRU-attention-based model achieved the smallest testing RMSE, showing improvements over the pure GRU-based model. The TCN-based model showed a similar performance as the GRU-attention-based model, followed by the GRU-based and Transformer-based models. The overall testing RMSEs for dataset 2 task were smaller than that for dataset 1 task, which was likely due to the significantly larger dataset size.

The testing results suggest that different architectures have unique strengths in addressing different tasks, and careful choices can achieve the most robust performance. Although the overall performances of different architectures are close to each other in our demonstrated examples of tasks, two potential scenarios should be kept in mind. Firstly, the strength/weakness of different architectures may be exemplified in real-world applications that encounter datasets of different sizes and even more complex stress-strain relations. Secondly, even small reductions in validation/testing errors may lead to qualitative improvements in capturing certain features, making model selection worthwhile.

Additionally, we need to consider the capability of the trained models in predicting unseen scenarios, both interpolation and extrapolation. In the following Sections, we demonstrate the applications of trained models in predicting unseen scenarios.

### 3.2. *Applications of dataset 1 models in unseen loading scenarios*

In this section, we present two applications of dataset 1 models to predict stress-strain behavior for unseen loading paths. The first loading path involves uniaxial loading from zero strain to final fracture strain without any unloading/reloading cycles. Although our models were trained on stress-strain curves with multiple unloading/reloading cycles, they should be able to predict a smooth strain hardening behavior in the unseen regions where unloadings were initiated, if they have learned the physical intuition of stress-strain relations.

Fig. 15 shows the predictions from various encoder-based models. It should be noted that the blue solid line represents the average prediction across five different models, and the shaded band indicates the uncertainty estimation based on the standard deviations of these predictions. All models were able to predict a smooth stress-strain curve, except for a few regions immediately after reloading, where predictions slightly deviated from the expected trend and showed relatively large uncertainties. The GRU-attention and Transformer encoder-based models performed better in these challenging regions. Additionally, the Transformer encoder-based models provided the most accurate predictions at the very beginning of elastic deformations. Overall, these predictions are quite satisfactory and demonstrate the models' capability of capturing the underlying physical relations.

The second loading path involves multiple unloading-reloading

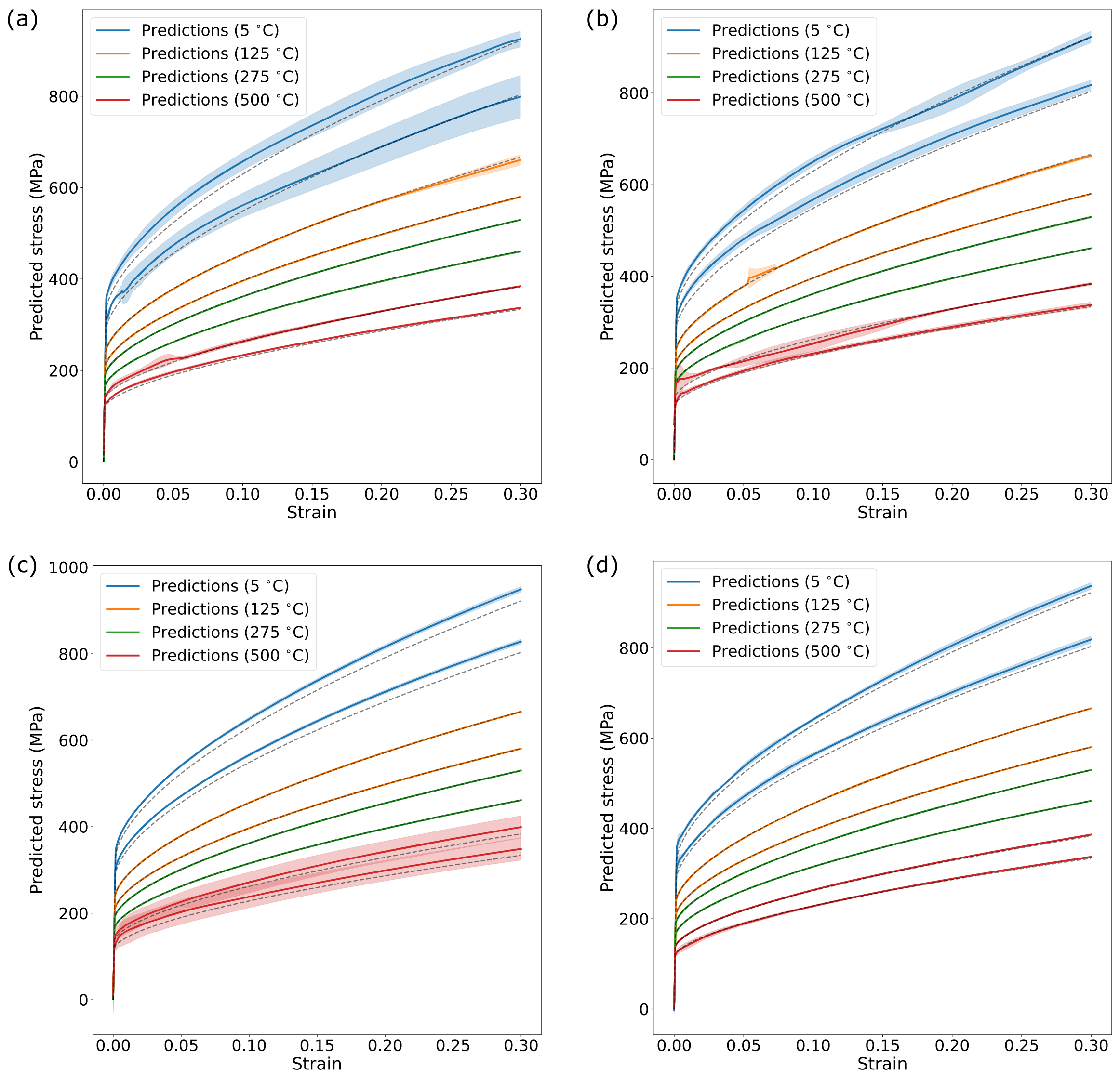

**Fig. 17.** Dataset 2 models' predictions on unseen loading scenarios. (a) Predictions from GRU-based models. (b) Predictions from GRU-attention-based models. (c) Predictions from TCN-based models. (d) Predictions from the Transformer encoder-based models. Four temperatures were considered, among which 5°C and 500°C are out-of-domain temperatures. For each temperature, two strain rates, $10^{-3}$ s$^{-1}$ and $10^{3}$s$^{-1}$, were tested. Dashed lines are the ground truth from Johnson-Cook model. Solid lines are the average prediction across five different models, while shaded bands indicate the uncertainty based on the standard deviations of the five models' predictions.

cycles that were not present in the training data. As demonstrated earlier, all models seem to have learned the underlying physical relations instead of just memorizing the training data pattern. Consequently, we expect the trained models to correctly predict the sharp transitions, elastic behaviors, and strain-hardening behaviors involved in unloading-reloading cycles at any point during the plastic deformation stage, including those not seen in the training dataset. To evaluate this, we introduced five unseen initiation points in the plastic deformation regime to complete unloading-reloading cycles. Fig. 16 shows various models' performance on this task. The GRU and Transformer encoder-based models both exhibit excellent predictions that adequately capture the sharp transitions, linearity of elastic deformations, and strain hardening behavior near the end of reloading. Although the GRU-attention and TCN encoder-based models showed relatively large uncertainties in certain regions, where the predicted curve was not smooth enough or deviated slightly from the expected shapes, they still captured the overall trend in a reasonable manner. These predictions on challenging unseen tasks suggest that the dataset-1 models indeed learned the underlying correlations between stress and strain for various loading scenarios.

### 3.3. Applications of dataset 2 models in unseen loading scenarios

In this dataset 2 task, we tested the models' performance on previously unseen loading conditions. We chose four unseen loading temperatures, including 5°C, 125°C, 275°C, and 500°C, and two unseen strain rates ($10^{-3}$ s$^{-1}$ and $10^{3}$ s$^{-1}$) at each temperature. Note that the temperatures 5°C and 500°C were out of the training temperature range, therefore they represent out-of-domain or extrapolation applications. Fig. 17 shows the performance of various models on these unseen loading conditions. All models performed excellently in predicting the stress-strain curves for temperatures 125°C and 275°C, with their predictions (solid curves) almost overlapping with the ground truth (dashed lines, from the Johnson-Cook model) and negligible uncertainties. Although these temperatures (125°C and 275°C) were unseen during training, they were contained in the training temperature range, thus suggesting excellent interpolation capability. For the out-of-domain or extrapolation applications at 5°C and 500°C, the Transformer encoder-based model exhibited the best performance, with its predicted curves nearly overlapping with the ground truth and negligible uncertainties. The GRU, GRU-attention, and TCN encoder-based models also showed reasonable overall predictions, with slight deviations in some regions from the ground truth, indicating relatively low bias. However, these models showed relatively high uncertainties in some regions, suggesting relatively high variance. Such relatively low bias and relatively high variance indicate some overfitting, which could be mitigated by reducing the models' complexity.

Our framework is not limited to loading conditions such as temperatures and strain rates; it can also include other input information, such as materials processing conditions and materials structural/chemical information. This makes the framework useful in materials optimization processes, such as in identifying the best processing conditions in additive manufacturing. By proper model selection and achieving a bias-variance balance, the trained model can be used to predict materials mechanical behavior under unseen processing conditions and provide uncertainty estimation. This predictive capability and uncertainty estimation can be used in search strategies, such as Bayesian optimization, to effectively search optimal processing conditions. This approach can significantly reduce experimental costs, reduce human bias, and accelerate materials development.

## Summary

In this study, we developed a deep learning framework for modeling constitutive relations under various conditions, using a general encoder-decoder architecture with different encoders such as gated recurrent unit (GRU), GRU with attention, temporal convolutional network (TCN), and Transformer. We tested and validated these models on two datasets with complex loading histories and various loading conditions. The optimal architectures with a root mean squared error converged below 1 MPa were selected for best performance in capturing various deformation behaviors. All selected architectures demonstrated excellent performance in capturing the underlying stress-strain relations, both in testing and out-of-domain scenarios. Based on the applications in this work, the Transformer encoder-based model demonstrated the best overall performance in both dataset tasks. Due to the universal model nature and excellent generalization ability, we expect the proposed framework to be applicable in a wide range of loading scenarios and mechanical behaviors such as fatigue loadings, hysteresis in shape memory alloys, etc. Taking advantage of the predictive capability and uncertainty estimation from deep ensembles, this framework can potentially be integrated into an active learning / Bayesian optimization [44] process for materials optimization, which can help reduce experimental cost, minimize human bias, and accelerate materials development (such as identifying optimal material processing/fabrication parameters in high throughput additive manufacturing [30,45]). Finally, we note that, in addition to robust model architectures, data consistency is also crucial for successful constitutive modeling, as different sources of stress-strain data often show uncontrolled variabilities [46].

## Declaration of Competing Interest

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

## Acknowledgment

This work was supported by the Laboratory Directed Research & Development Program at Idaho National Laboratory under the Department of Energy (DOE) Idaho Operations Office (an agency of the U.S. Government) Contract DE-AC07-05ID145142. This research made use of Idaho National Laboratory computing resources which are supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517. J. Li and Q-J. Li also acknowledge support by NSF CMMI-1922206.

## References

[1] J.H. Sung, J.H. Kim, R.H. Wagoner, A plastic constitutive equation incorporating strain, strain-rate, and temperature, Int. J. Plast. 26 (2010) 1746–1771, https://doi.org/10.1016/j.ijplas.2010.02.005.
[2] E. VOCE, The Relationship between Stress and Strain for Homogeneous Deformation, J. Inst. Met. 74 (1948) 537–562.
[3] G.R. Johnson, W.H. Cook, A constitutive model and data for metals subjected to large strains, high strain rates and high temperatures, in: Proceedings of the 7th International Symposium on Ballistics, The Hague, The Netherlands, 1983, pp. 541–547.
[4] F.J. Zerilli, R.W. Armstrong, Dislocation-mechanics-based constitutive relations for material dynamics calculations, J. Appl. Phys. 61 (1987) 1816–1825, https://doi.org/10.1063/1.338024.
[5] S.R. Bodner, Y. Partom, Constitutive equations for elastic-viscoplastic strain-hardening materials, J. Appl. Mech. 42 (1975) 385–389, https://doi.org/10.1115/1.3423586.
[6] A.S. Khan, S. Huang, Experimental and theoretical study of mechanical behavior of 1100 aluminum in the strain rate range 10−5−104s−1, Int. J. Plast. 8 (1992) 397–424, https://doi.org/10.1016/0749-6419(92)90057-J.
[7] W. Ramberg, W.R. Osgood, W.R. Osgood, Description of stress-strain curves by three parameters, (1943). https://ntrs.nasa.gov/citations/19930081614 (accessed March 28, 2022).
[8] K.J.R. Rasmussen, Full-range stress–strain curves for stainless steel alloys, J. Constr. Steel Res. 59 (2003) 47–61, https://doi.org/10.1016/S0143-974X(02)00018-4.
[9] P. Ludwik, Elemente der Technologischen Mechanik, Springer, Berlin Heidelberg, 1909.
[10] J.H. Hollomon, Tensile deformation, Trans. Metall. Soc. AIME 162 (1945) 268–290.
[11] R. Liang, A.S. Khan, A critical review of experimental results and constitutive models for BCC and FCC metals over a wide range of strain rates and temperatures, Int. J. Plast. 15 (1999) 963–980, https://doi.org/10.1016/S0749-6419(99)00021-2.
[12] J.L. Chaboche, Constitutive equations for cyclic plasticity and cyclic viscoplasticity, Int. J. Plast. 5 (1989) 247–302, https://doi.org/10.1016/0749-6419(89)90015-6.
[13] F. Mollica, K.R. Rajagopal, A.R. Srinivasa, The inelastic behavior of metals subject to loading reversal, Int. J. Plast. 17 (2001) 1119–1146, https://doi.org/10.1016/S0749-6419(00)00082-6.
[14] E.F. Rauch, J.J. Gracio, F. Barlat, Work-hardening model for polycrystalline metals under strain reversal at large strains, Acta Mater. 55 (2007) 2939–2948, https://doi.org/10.1016/j.actamat.2007.01.003.
[15] X. Ling, T. Belytschko, Thermal softening induced plastic instability in rate-dependent materials, J. Mech. Phys. Solids 57 (2009) 788–802, https://doi.org/10.1016/j.jmps.2008.04.010.
[16] F. Barlat, J. Ha, J.J. Grácio, M.-.G. Lee, E.F. Rauch, G. Vincze, Extension of homogeneous anisotropic hardening model to cross-loading with latent effects, Int. J. Plast. 46 (2013) 130–142, https://doi.org/10.1016/j.ijplas.2012.07.002.
[17] T.N. Nguyen, T. Siegmund, V. Tomar, J.J. Kruzic, Interaction of rate- and size-effect using a dislocation density based strain gradient viscoplasticity model, J. Mech. Phys. Solids 109 (2017) 1–21, https://doi.org/10.1016/j.jmps.2017.07.022.
[18] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signal Syst. 2 (1989) 303–314, https://doi.org/10.1007/BF02551274.
[19] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (1991) 251–257, https://doi.org/10.1016/0893-6080(91)90009-T.
[20] J. Ghaboussi, J.H. Garrett, X. Wu, Knowledge-based modeling of material behavior with neural networks, J. Eng. Mech. 117 (1991) 132–153, https://doi.org/10.1061/(ASCE)0733-9399(1991)117:1(132).
[21] J. Ghaboussi, D.A. Pecknold, M. Zhang, R.M. Haj-Ali, Autoprogressive training of neural network constitutive models, Int. J. Numer. Methods Eng. 42 (1998) 105–126, https://doi.org/10.1002/(SICI)1097-0207(19980515)42:1<105::AID-NME356>3.0.CO;2-V.
[22] M. Lefik, B.A. Schrefler, Artificial neural network as an incremental non-linear constitutive model for a finite element code, Comput. Methods Appl. Mech. Eng. 192 (2003) 3265–3283, https://doi.org/10.1016/S0045-7825(03)00350-5.
[23] Y.M.A. Hashash, S. Jung, J. Ghaboussi, Numerical implementation of a neural network based material model in finite element analysis, Int. J. Numer. Methods Eng. 59 (2004) 989–1005, https://doi.org/10.1002/nme.905.
[24] S. Jung, J. Ghaboussi, Neural network constitutive model for rate-dependent materials, Comput. Struct. 84 (2006) 955–963, https://doi.org/10.1016/j.compstruc.2006.02.015.
[25] M.B. Gorji, M. Mozaffar, J.N. Heidenreich, J. Cao, D. Mohr, On the potential of recurrent neural networks for modeling path dependent plasticity, J. Mech. Phys. Solids 143 (2020), 103972, https://doi.org/10.1016/j.jmps.2020.103972.
[26] K. Xu, A.M. Tartakovsky, J. Burghardt, E. Darve, Learning viscoelasticity models from indirect data using deep neural networks, Comput. Methods Appl. Mech. Eng. 387 (2021), 114124, https://doi.org/10.1016/j.cma.2021.114124.

[27] Y. Zhang, Q.-J. Li, T. Zhu, J. Li, Learning constitutive relations of plasticity using neural networks and full-field data, Extreme Mech. Lett. 52 (2022), 101645, https://doi.org/10.1016/j.eml.2022.101645.
[28] X. Li, C.C. Roth, D. Mohr, Machine-learning based temperature- and rate-dependent plasticity model: application to analysis of fracture experiments on DP steel, Int. J. Plast. 118 (2019) 320–344, https://doi.org/10.1016/j.ijplas.2019.02.012.
[29] K. Linka, M. Hillgärtner, K.P. Abdolazizi, R.C. Aydin, M. Itskov, C.J. Cyron, Constitutive artificial neural networks: a fast and general approach to predictive data-driven constitutive modeling by deep learning, J. Comput. Phys. 429 (2021), 110010, https://doi.org/10.1016/j.jcp.2020.110010.
[30] N.M. Heckman, T.A. Ivanoff, A.M. Roach, B.H. Jared, D.J. Tung, H.J. Brown-Shaklee, T. Huber, D.J. Saiz, J.R. Koepke, J.M. Rodelas, J.D. Madison, B. C. Salzbrenner, L.P. Swiler, R.E. Jones, B.L. Boyce, Automated high-throughput tensile testing reveals stochastic process parameter sensitivity, Mater. Sci. Eng.: A 772 (2020), 138632, https://doi.org/10.1016/j.msea.2019.138632.
[31] K. Cho, B. van Merriënboer, 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), Doha, Qatar, Association for Computational Linguistics, 2014, pp. 1724–1734, https://doi.org/10.3115/v1/D14-1179.
[32] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, ArXiv:1803.01271 [Cs]. (2018). http://arxiv.org/abs/1803.01271 (accessed October 21, 2021).
[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need. Advances in Neural Information Processing Systems, Curran Associates, Inc., 2017, in: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. accessed February 14, 2023.
[34] D. Umbrello, R. M'Saoubi, J.C. Outeiro, The influence of Johnson–Cook material constants on finite element simulation of machining of AISI 316L steel, Int. J. Mach. Tools Manuf 47 (2007) 462–470, https://doi.org/10.1016/j.ijmachtools.2006.06.006.
[35] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in Neural Information Processing Systems, Curran Associates, Inc., 2017, in: https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html. accessed October 2, 2022.
[36] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.
[37] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, (2016). https://doi.org/10.48550/arXiv.1409.0473.
[38] J. Ghaboussi, D.E. Sidarta, New nested adaptive neural networks (NANN) for constitutive modeling, Comput. Geotech. 22 (1998) 29–52, https://doi.org/10.1016/S0266-352X(97)00034-7.
[39] M.N. Cinbiz, The effect of stress state on zirconium hydride reorientation, 2015. https://ui.adsabs.harvard.edu/abs/2015PhDT.......286C (accessed July 27, 2022).
[40] M.N. Cinbiz, D.A. Koss, A.T. Motta, The influence of stress state on the reorientation of hydrides in a zirconium alloy, J. Nucl. Mater. 477 (2016) 157–164, https://doi.org/10.1016/j.jnucmat.2016.05.013.
[41] O.N. Pierron, D.A. Koss, A.T. Motta, Tensile specimen geometry and the constitutive behavior of Zircaloy-4, J. Nucl. Mater. 312 (2003) 257–261, https://doi.org/10.1016/S0022-3115(02)01554-4.
[42] ASTM E8 /E8M-21, Standard Test Methods for Tension Testing of Metallic Materials, ASTM International, West Conshohocken, 2021.
[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: an imperative style, high-performance deep learning library, ArXiv:1912.01703 [Cs, Stat]. (2019). http://arxiv.org/abs/1912.01703 (accessed May 3, 2022).
[44] D. Morgan, G. Pilania, A. Couet, B.P. Uberuaga, C. Sun, J. Li, Machine learning in nuclear materials research, Curr. Opin. Solid State Mater. Sci. 26 (2022), 100975, https://doi.org/10.1016/j.cossms.2021.100975.
[45] C. Sun, Y. Wang, M.D. McMurtrey, N.D. Jerred, F. Liou, J. Li, Additive manufacturing for energy: a review, Appl. Energy 282 (2021), 116041, https://doi.org/10.1016/j.apenergy.2020.116041.
[46] B.S. Aakash, J. Connors, M.D. Shields, Variability in the thermo-mechanical behavior of structural aluminum, Thin Walled Struct. 144 (2019), 106122, https://doi.org/10.1016/j.tws.2019.01.053.