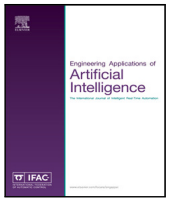
Engineering Applications of Artificial Intelligence 137 (2024) 109165

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Research paper

Time mesh independent framework for learning materials constitutive relationships

Marcello Laurenti [a], Qing-Jie Li [b], Ju Li [b,c,*]

[a] Department of Chemical Engineering Materials Environment, Sapienza-Università di Roma, Via Eudossiana 18, Rome, 00184, RM, Italy
[b] Department of Nuclear Science and Engineering, Massachusetts Institute of Technology, Cambridge, 02139, MA, United States
[c] Department of Material Science and Engineering, Massachusetts Institute of Technology, Cambridge, 02139, MA, United States

ARTICLE INFO

Keywords:
Bio-inspired framework
Constitutive behavior
Uneven and noise data
Tensile testing predictions
Robustness against data unevenness

ABSTRACT

Real-world datasets are rarely populated by evenly distributed entries; unevenness may be caused by sensor malfunctions or randomized sampling due to the process nature. Modeling the constitutive relationship (CR) of materials in scenarios where the temporal data available are uneven is a serious challenge for black box approaches such as artificial neural networks. This work presents a general framework capable of modeling uneven sampled data, which is composed of an Encoder–Decoder (ED) structure. In our framework, the Encoder can process an uneven input sequence, thanks to an approximation of the Ordinary Differential Equations (ODE), and project it into a lower dimensional latent space; the Decoder, on the other hand, can map the compressed information into the output of interest, the material stress response in this work. In the proposed temporal mesh independent framework, the Encoder is a multi-layer structure, with each layer consisting of a Long-Short Term Memory (LSTM) layer, a Closed form Continuous Time (CfC) layer, and a Self Multi-Head Attention Layer (MHAL) layer connected in series. The Decoder can be one Fully Connected Network (FCN) or two FCNs in parallel; in the latter case, the Decoder is capable of giving the mean and the variance of the output. The presented mesh-independent framework demonstrates good accuracy despite both the unevenness and the noise of the training data, specially when its results are compared to the standard ones; thus extending the applicability of neural-network-based black box models in real world applications.

1. Introduction

Material properties and responses under certain conditions have long been modeled using the so-called “white box models”, a category of mathematical models with well-motivated analytical relations. However, such an approach requires a good understanding of the physical process or property being studied. For processes/properties that are difficult to understand or too complex to model with simple mathematical formulations, the so-called “black box model” approach is often adopted, which can be viewed as an evolution of the empirical or phenomenological approach. In this case, the relations being modeled can be nonlinear and in a high-dimensional space, e.g., thousands of variables may be transformed in ways that we humans are unable to perceive straightforwardly, thus the name “black box”. Noteworthy examples of this kind of approach include various Machine Learning methods and Artificial Neural Networks.

Advances in computational hardware and neural network (NN) architectures have spurred extensive interest in NN modeling of materials properties under complex conditions. In the last decade, several research have successfully deployed Deep NNs to model a variety of different material properties and behaviors. Li et al. (2023b) predicted the mechanical properties of carbon fibers (CF) by using a NN composed by two heads, a Convolutional NN (CNN) and a Multi Layer Perceptron (MLP), which were given as input the visual and the contextual text information respectively. In their work, they achieved promising accuracy on different CF’s mechanical properties with an R^2-value of 0.99, highlighting the architecture’s capability of predicting material properties via multi-source heterogeneous data. Gholami et al. (2023) achieved great accuracy deploying residual networks such as ResNet (He et al., 2015) or AlexNet (Krizhevsky et al., 2012) to predict the mechanical properties of a bio-glass (BG) collagen (COG) composite. In their work, both NNs’ hyperparameters were tuned to ensure great regression performances with R^2-values around 0.99. Ning et al. (2023) successfully used a Harris Hawks Optimization (HHO) algorithm (Hussien et al., 2019) coupled with a Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) to accurately predict the Remaining Useful Life (RUL) of super-capacitors

* Corresponding author at: Department of Nuclear Science and Engineering, Massachusetts Institute of Technology, Cambridge, 02139, MA, United States.
Linkedin: marcello-laurenti-a541811b7 (M. Laurenti).
E-mail addresses: marcello.laurenti@uniroma1.it (M. Laurenti), qingjie.li219@gmail.com (Q.-J. Li), liju@mit.edu (J. Li).

https://doi.org/10.1016/j.engappai.2024.109165
Received 4 October 2023; Received in revised form 29 May 2024; Accepted 17 August 2024
Available online 6 September 2024

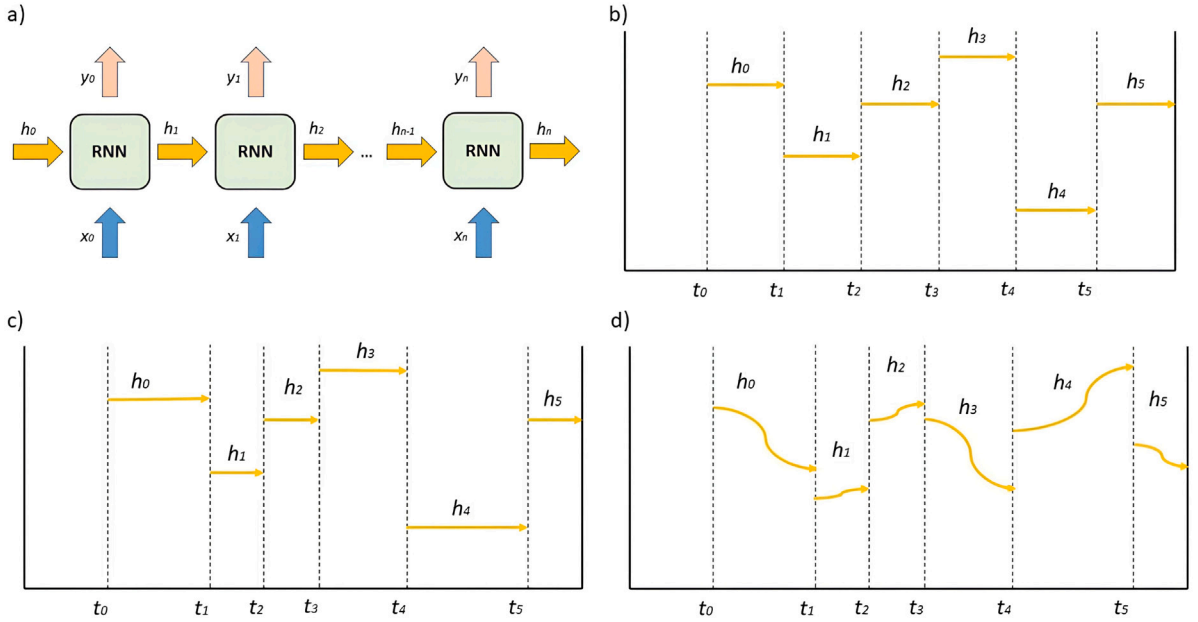


Fig. 1. (a) A schematic of the information flow in a generic RNN Cell. At each time t, the input x_t and the h_{t-1} hidden state are used to compute the output and the next hidden state h_t; (b) When input points are equally distant from each other the information flows correctly from $t-1$ to t and the network is able to learn data patterns; (c) When input points are unevenly distributed, the information flow does not account for the different Δt, thus causing poor performances; (d) When the hidden states are a function of Δt, the information flows and it accounts the differences in Δt.

under different temperature settings. Marco et al. (2021) used a network composed of a CNN and an MLP to assess, from a 2D image, the interface interlocking type and its mechanical properties such as stiffness, ultimate tensile strength and toughness. Zheng et al. (2018) used a CNN network to learn, from the information contained in the periodic table, both the lattice parameter and enthalpy of formation of a compound simultaneously. In another similar work, Dai et al. (2023) developed a highly accurate Graph NN to efficiently correlate the crystalline structure of poly-crystalline materials with their Li-ion 3-dimensional conductivity, outperforming a linear regression model and two baseline convolutional NN models. Hestroffer et al. (2023) used state-of-the-art graph NN(GNN)-3D-CNN to map polycrystalline features such as crystallographic orientation, size, and grain neighbor connectivity information to their relative mechanical properties. In their work the AI was capable of predicting the mechanical properties associated with the polycrystalline structures with a mean absolute error (MAE) lower than 1%.

As stress–strain responses are usually dependent on load histories, neural network (NN) architectures capable of modeling temporal sequences have been widely explored. Marco et al. (2022) used Graph NNs to predict strain, stress and deformation in various material systems, like fiber and stratified composites, and lattice meta-materials. In the same sub-field, but working with more complex displacement fields, Gorji Maysam et al. (2020) developed a Recurrent NN (RNN)-based framework to model the large deformation response of elasto-plastic solids subject to arbitrary multi-axial loading paths, where the scalar time is used to parameterize both stress and strain. Recently Li et al. (2023a) extended the RNN-based architecture into a general encoder–decoder framework, where the encoder can be any sequence modeling NNs such as Gated Recurrent Unit (GRU (Cho et al., 2014)), LSTM, Temporal Convolutional Network (TCN), and Transformer, etc. to project the high dimensional loading sequence data onto a compact latent space, and the Decoder subsequently maps the encoded information to stress responses. Such a flexible framework demonstrated promising results for materials under complex loading conditions.

Although such NN frameworks can learn the dynamics of the underlying processes, like an animal can learn how to grab onto tree branches and swing from treetop to treetop, they do not integrate explicit knowledge of all the underlying physics, and thus posing several limitations. For example, these frameworks have not been demonstrated for modeling loading data with uneven/arbitrary sampling intervals, which however is important in practical applications.

The predictive capabilities of a RNN-based model depend on how the input sequence is processed and how the states are passed from t_i to t_{i+1}. A visual representation of this is shown in Fig. 1a. Besides how the cell states are computed, they are simply passed to the next unit to compute the t_{i+1} prediction and cell states. This *modus operandi* assumes some level of *symmetry* (Fig. 1b) in the input data, for example if the inputs are time-dependent, so it can be expressed as a function $X(t)$, the network implicitly assumes that between the couple $X(t_i)$, $X(t_{i+1})$ and $X(t_{i+1})$, $X(t_{i+2})$, the time differences $t_{i+1} - t_i$ and $t_{i+2} - t_{i+1}$ are (nearly) constant Δt for all the input couples in the dataset. This assumption is true in a great number of the data structures available, for example performing a Strain-Stress test on a given material, the sample rate of both displacement and force is often maintained constant throughout the test. With such pattern, architectures like GRU or LSTM are able to learn the data structure and can give accurate predictions. Such "black box" models are explored in more detail in the work done by Li et al. (2023a). Problems start to arise when this temporal mesh symmetry is absent (Fig. 1c), because, in their inner structure, GRU (Cho et al., 2014) and LSTM (Hochreiter and Schmidhuber, 1997) are unable to model input unevenness. This leads to concerns about the NN capability to make accurate predictions under more complex time-sampling conditions.

Unevenness and noise in temporal data sampling are generally present due to factors such as sensor malfunctions, electrical blackouts, sensor resolutions or a nonlinear sampling rate like in a creep test scenario. Such irregularly sampled or uneven temporal datasets still hold very useful information, but they are hardly learnable by standard black box models (Fig. 2a), thus limiting their applications in many fields. In this context, having a general framework more robust against time unevenness, or generally data unevenness, is crucial to maximizing data utilization and information extraction, especially if data collection is expensive. Being capable of modeling uneven data, both in time

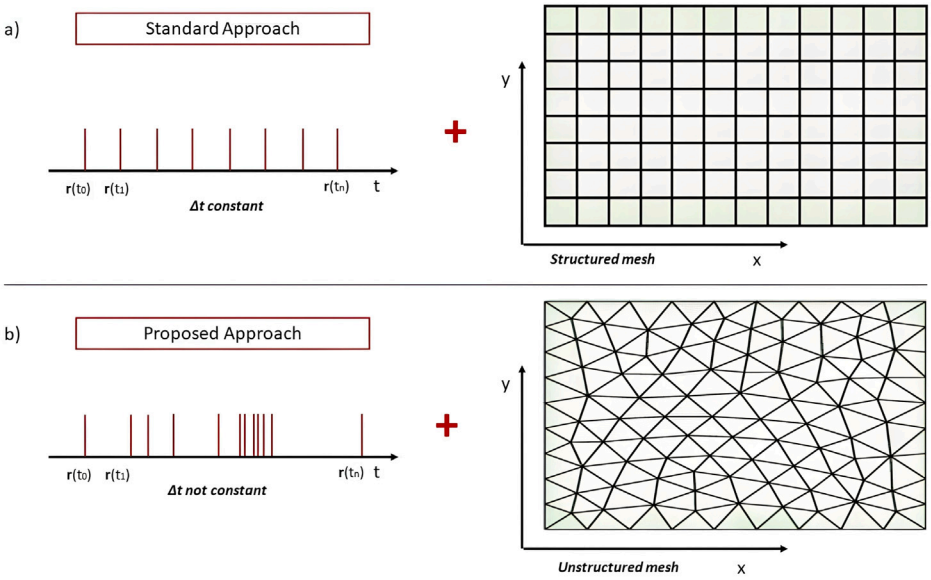


Fig. 2. Differences between standard black box approach (a) and proposed black box approach (b). In this work data entries are severely unevenly distributed both in space and time. In a standard application, the observation will still conserve some degree of time symmetry.

and space (Fig. 2b) (or other dimensions) would promote the black model approach as robust as the white box models used in ODE solvers with adaptive (thus uneven) time-stepping or as finite element method (FEM) solvers for partial differential equations (PDE) that use adaptive spatial and temporal meshes, with the advantage of predicting the system behavior without necessarily knowing its true dynamics in closed form.

Several approaches have been explored to tackle this issue. Baytas et al. (2017) tried to treat Δt as a feature in the input sequence, but their results showed little improvement with an accuracy comparable to the standard RNNs structures. As time is used as an additional input dimension to a standard RNN, this architecture lacks an inner structure capable of approximating the time dependencies. Che et al. (2018), instead, tried to model the cell state as a function of Δt, c(Δt), using an exponential decay function, however, this approach showed no improvement when the Δt between observations was very large due to excessive decay. While an exponentially time decaying effect could be beneficial in decorrelating two temporally-distant hidden states when the Δt are relatively small, it causes a complete destruction of the information flow when Δt are larger; thus this methodology is not adequate to model *true* data unevenness. A big leap into addressing the time-invariance problem was proposed by Chen et al. (2018) with the introduction of Neural Ordinary Differential Equations (NODE). Their work clearly showed how the cell state is a function of time and varies accordingly to the Δt among observations (Fig. 1d) and this is reflected in improved accuracy with respect to any normal RNN networks. The major limitation of this newly proposed architecture was the excessive computation time and the added memory required to run. This was due to the nonlinear relation between input and output, the exponential functions involved, and the computation of an ODE. Furthermore, the presence of more variables to optimize during the optimization process limited the application of this architecture to small datasets or simple networks in general. An improvement to such structures was made by Hasani et al. (2022), with the development of the CfC structure, an acronym that stands for Closed Form Continuous Time. CfC is a particular approximation of the Liquid Time Constant Networks done by Hasani et al. (2020) that can more effectively correlate observations taken with different Δt. The network proposed is a variant of a standard RNN with both an inner gating mechanism and a linear time handling. Even though this architecture is less rigorous than NODE, it is faster and requires fewer parameters, thus retaining good time-handling capabilities.

Lechner et al. also developed a particular synapse wiring called NCP (Lechner et al., 2020) for the CfC Network based on the *Caenorhabditis Elegans* nervous system (Fig. 3). NCP showed two advantages compared to the usual fully connected NN. First, the wiring in NCP assumes that not every connection is needed to compute the output (in contrary to a standard Fully Connected Network). A MLP is built by connecting every neuron on the previous and next layer, however, not every connection is needed to successfully approximate the hyper-surface associated with the task, so during the optimization process, many coefficients are zeroed. When the network has a simple structure, zeroing coefficients is not a difficult task for the optimization algorithm; however, when the network grows in complexity, the added dimensionality creates a more complex hyper-surface and increases the risk for the optimization process to get permanently stuck in a local minimum, thus increasing the error in the prediction. In NCP networks, during the initialization, some coefficients are permanently zeroed, thus they do not participate in the gradient descent algorithm. The NCP network can retain a higher latent space without compromising the optimization process by having fewer parameters to optimize, thus allowing the modeling of more complex trajectories. The second advantage of NCP is obtained by using this sparse connection policy on three different layers, mimicking the real neural connections of a living organism such as *Caenorhabditis Elegans*.

Although such structure can efficiently correlate hidden states with different Δt, they can suffer from a phenomenum called "data memorization". NNs are usually composed of millions, if not billions, of optimizable parameters, but they are used to fit a relatively smaller number of observations from training dataset; which can lead to a *memorization* of the training data distribution. This effect differs from the *standard overfit* or *memorization* that is indicated by the divergence between training and testing errors. The above-mentioned *distribution memorization* causes the performances to drop significantly when slightly different data distribution, or noise, is used during *inference* of a trained model. This effect can be also interpreted as a form of *network poisoning*, and it is more evident in NNs such as Image Classifiers. During the last years different independent researchers pointed out how brittle Image Classifiers are against simple attacks, such as One Pixel Attach (Su et al., 2019; Alatalo et al., 2022), Patch Attack (Sen

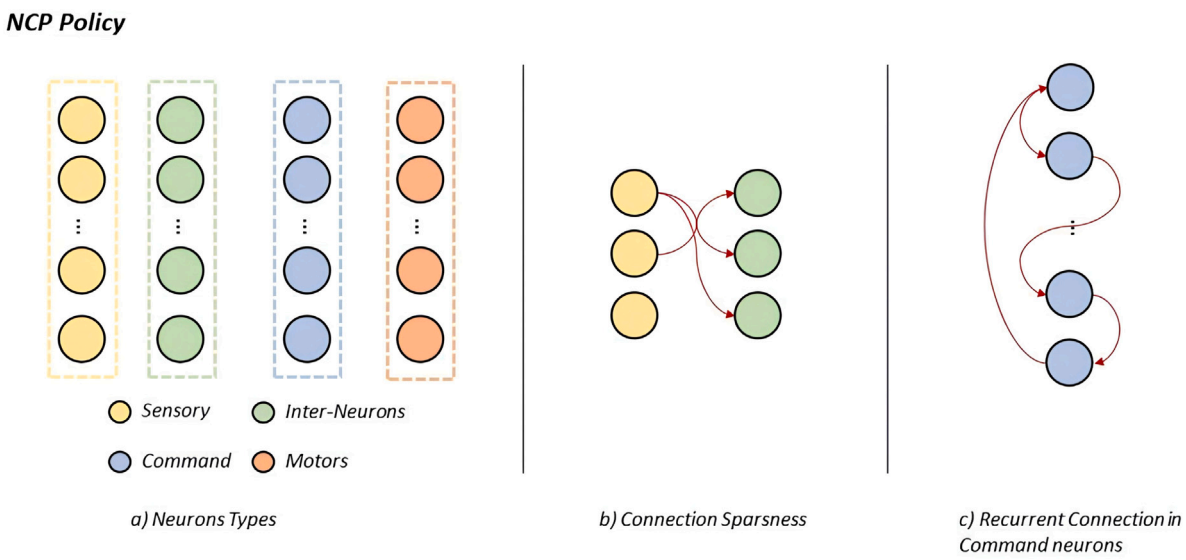


Fig. 3. (a) Schematic representation of the type of neurons involved in the NCP. Sensory neurons are the source of information in a R^n space, where n is the number of features that are carried. Inter-Neurons, Command Neurons and Motor Neurons are the proper neurons of the structure, characterized by their own synaptic connections, weight and biases; (b) Connections in this structure are sparse, so not every neuron is connected to each other; (c) The Command Neurons layer has both sparseness in the synaptic connection between neurons of other layers and sparseness in the recurrent connection between neurons of the same layer.

and Dasgupta, 2023) and Noise Attack (Duan et al., 2021; Sen and Dasgupta, 2023). In those works, small changes, imperceptible to us humans, have completely altered the prediction of otherwise very accurate Image Classifiers.

Such brittleness extends to other types of neural architecture and it could have very undesirable effects in Material Science. Small changes in the data could potentially compromise the accuracy of an otherwise well-tuned NN; thus possibly leading to catastrophic predictions if the NNs are used in real-world scenarios. The above-mentioned changes can easily permeate inside the data-gathering workflow, and they have two different sources:

- *Different sampling procedure*:
  - A sensor *dissimilar* from the others previously used to populate the dataset could be introduced into the data-gathering flow. The new observations, even if they seem coherent and identical to the others, could introduce noise or skewness into the dataset;
  - The sampling procedure could be changed, thus the observations could be not equidistant between each other. It could be, for example, a decision to change the frequency of measurement;
  - A researcher or an operator could decide to apply the model on a different data distribution from the training one. For example, a different sampling interval or noises may be used during model inference.
- *Different testing environment*: a NN could be deployed in the real world after being successfully trained in a simulated environment. The new *real* observations would have different distributions from the *simulated* ones and the model prediction could suffer a severe decrease in accuracy.

In Materials Science field, observations are usually not cheap both economic-wise and time-wise; thus this brittleness poses a serious limitation to the use of deep learning modeling.

The ultimate goal of this research is to develop a robust framework against data sparseness and noise, capable of achieving accurate results in environments characterized by a broken time-symmetry. Such scenarios usually cause the non convergence of the NNs or significantly decrease the model accuracy in *inference mode*. A secondary but equally important goal of this research is to also achieve equal accuracy in the aforementioned *inference mode* when the data distribution differs from the training and test datasets. In this work, we achieved these goals by using an Encoder–Decoder architecture composed of several CfC Layers wired with NCP, each coupled with a Multi Head Attention Layer (MHAL)(Fig. 3b, c), extending the network robustness against data unevenness. In our proposed network the CfC layers are employed to link uneven and different data distributions using the Δt_i between observations. The framework has proven to be capable of giving accurate and reasonable predictions even when highly damaged datasets with multiple missing values as training data and different point distributions in *inference mode* have been used. Similar to the architecture by Li et al. (2023a), we further modified the Decoder as a dual parallel FCN to model both the mean and variance of the prediction (learning data distribution). This approach is particularly useful in cases such as mechanical testing data on the same material showing variations (e.g., from different manufacturers), or when input data are particularly noisy. Using a Variational Approach allows to introduce a *Confidence Region* for the predicted output, jointly adding further robustness to both noise and redundant data and giving interpretability to the network.

2. Methods

2.1. Learning tasks

In this research, the practical approach followed is very similar to the work done by Li et al. (2023a), but entails some substantial differences. Inputs and outputs are functions of time, so in the case of a strain-stress curve, the mechanical response to an applied strain must be intended as:

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

Where NN is the neural network to be learned, $\boldsymbol{\varepsilon}$ is the strain tensor that represents the field of deformations applied to the material, $\boldsymbol{\sigma}$ is the stress tensor, $\boldsymbol{\psi}$ is the tensor of auxiliary information that includes all the other conditions applied to the material or information

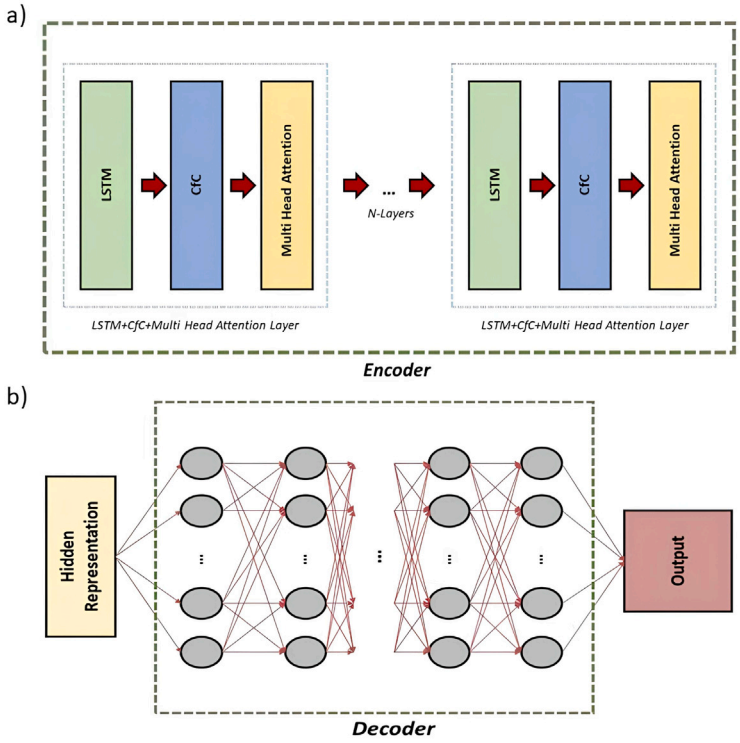


Fig. 4. (a) Encoder Structure, composed of N-layers of a series of LSTM, CfC wired with NCP and a Self Multi-Head Attention Mechanism; (b) Decoder Structure, composed of M layers of a FCN. The LSTM cells, thanks to the *cell state*, will learn the long-term dependencies in the train datasets; the CfC cells will use the Δt to correlate uneven observations. Jointly LSTM and CfC will learn useful and meaningful patterns from the sequence data, projecting them into a compact *latent space*. The retrieved information is then used by the MLP to learn seasonal and periodic patterns as well as trend, to give accurate predictions.

about the material fabrication, such as temperature, strain rate, angle, composition, etc. $\boldsymbol{\sigma}$ expanded in its components, can be written as:

$$\boldsymbol{\sigma}(t) = \sigma_1(t), \sigma_2(t), \sigma_3(t), \sigma_4(t), \sigma_5(t), \sigma_6(t) \tag{2}$$

$\boldsymbol{\epsilon}$ can also be expanded in its components, so it can be written as:

$$\boldsymbol{\epsilon}(t) = \varepsilon_1(t), \varepsilon_2(t), \varepsilon_3(t), \varepsilon_4(t), \varepsilon_5(t), \varepsilon_6(t) \tag{3}$$

In this study, the strain-stress curves examined were taken under a uniaxial load condition, so the latter relation simplifies in:

$$\sigma(t) = \mathrm{NN}[\varepsilon(t), \boldsymbol{\psi}(t)] \tag{4}$$

Using a RNN approach, a sequence of input data of length n must be provided as input. This can be represented as:

$$\boldsymbol{s}_t = [s_{t-n}, s_{t-n+1}, \dots, s_t] \tag{5}$$

In this context, each s_i is composed as:

$$s_i = [\varepsilon_i, \Delta t_i, \boldsymbol{\psi}_i] \tag{6}$$

Where Δt_i is the difference between the observation time of s_i and s_{i-1}. A RNN is particularly feasible for a learning task such as the one presented here because it can efficiently data-mine structures in the training database and because it is generally more robust, so the predictions are more stable and less subject to noise.

In the current work, an architecture that can be described as an Encoder–Decoder is used. The structure is composed of two main blocks:

- An **Encoder** (Fig. 4a) block that has the role to process the input sequences to a latent R^l subspace, effectively reducing data dimension while keeping essential information.
- A **Decoder** (Fig. 4b) block that processes the compressed information, learns periodic or seasonal cycles, and computes the output.

Compressing high dimensional input data to compact representation in the latent space, or in general, information compression is a key element to achieving not only efficient memory usage but also good predictive performance. This structure has also been found in the brains of high organisms (Motiwala et al., 2022), indicating its evolutionary advantage for the high intelligence of some species.

2.2. Framework structure: Encoder

The Encoder part in the proposed framework plays a very important role in processing unevenly spaced data. Each Encoder layer is composed of three different sub-blocks, each one with a very specific function that contributes to the efficiency of the network.

2.2.1. Encoder sub-block: LSTM

The first component in a single encoder layer is an LSTM NN (Hochreiter and Schmidhuber, 1997), a particular RNN (Rumelhart and McClelland, 1987) equipped with several gating mechanisms that allow remembering both recent and distant sequence information. Here the LSTM NN is used to properly model long-time dependencies. An LSTM NN structure has two distinct internal states:

- *hidden state* (h).
- *cell state* (c).

At time t, both are functions (F) of the input at time t, x_t, and their previous states at time $t-1$.

$$h_t = F[x_t, h_{t-1}, c_{t-1}] \tag{7}$$

$$c_t = F[x_t, h_{t-1}, c_{t-1}] \tag{8}$$

A schematic of its structure is shown in Fig. 5.

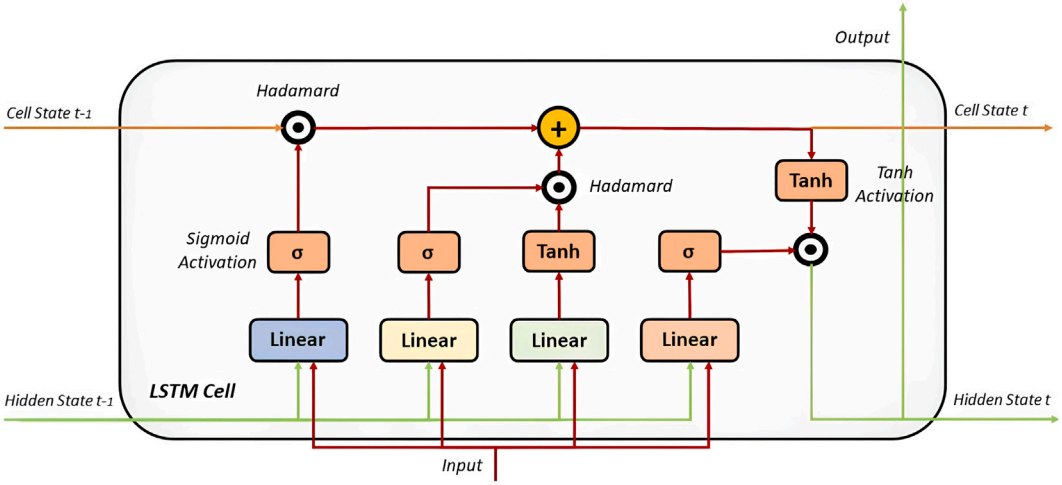


Fig. 5. Schematic representation of an LSTM Cell, which has four gating mechanisms: *input*, *cell*, *forget* and *output* gate. In the figure, *Linear* represents a linear projection layer, σ represents the Sigmoid activation function, *Tanh* represents the Hyperbolic Tangent activation function and $\odot$ represents the Hadamard product.

The key in this structure is the horizontal line represented by the *cell state*. It has very few interactions with everything else, and it can easily pass through without any major changes. It has only two interactions with the entering data, represented by a *Hadamard* product and a *Sum*. The first one is a pointwise product with the *forget gate*, a gating mechanism that decides how much past cell information should be discarded. If zero, everything is discarded; if one, everything is retained. The *forget gate* is computed as:

$$f_g = \sigma(\boldsymbol{W}_{if} \cdot x_i + \boldsymbol{b}_{if} + \boldsymbol{W}_{hf} \cdot h_{i-1} + \boldsymbol{b}_{hf}) \tag{9}$$

$\boldsymbol{\sigma}$ is the Sigmoid activation function; $\boldsymbol{W}_{if}$ and $\boldsymbol{W}_{hf}$ are the matrices storing the forget gate projection weights for input and hidden state, respectively; $\boldsymbol{b}_{if}$ and $\boldsymbol{b}_{hf}$ are the tensors storing the forget gate biases for input and hidden state, respectively; x_i is the i_{th} trajectory point and h_{i-1} is the previous hidden state.

The *Sum* is responsible for adding new information to the internal cell state. The mechanism that decides which information could be passed through is decided by a two-step operation. The first step is represented by the computation of the *input gate*, which, aside from the coefficients involved, has the same form as the forget gate, but this one is responsible for deciding which information can be uploaded. The *input gate* is computed as:

$$i_g = \sigma(\boldsymbol{W}_{ii} \cdot x_i + \boldsymbol{b}_{ii} + \boldsymbol{W}_{hi} \cdot h_{i-1} + \boldsymbol{b}_{hi}) \tag{10}$$

$\boldsymbol{W}_{ii}$ and $\boldsymbol{W}_{hi}$ are the matrices storing the input gate projection weights for input and hidden state, respectively; $\boldsymbol{b}_{ii}$ and $\boldsymbol{b}_{hi}$ are the tensors storing the input gate biases for input and hidden state, respectively.

The second step creates the $\tilde{c}$ *new candidate* values that can be uploaded to the *cell state*.

$$\tilde{c} = \tanh(\boldsymbol{W}_{ic} \cdot x_i + \boldsymbol{b}_{ic} + \boldsymbol{W}_{hc} \cdot h_{i-1} + \boldsymbol{b}_{hc}) \tag{11}$$

$\boldsymbol{W}_{ic}$ and $\boldsymbol{W}_{hc}$ are the matrices storing the candidates projection weights for input and hidden state, respectively; $\boldsymbol{b}_{ic}$ and $\boldsymbol{b}_{hc}$ are the tensors storing the candidates biases for input and hidden state, respectively.

The *new cell state* is then computed as:

$$c_t = c_{t-1} \odot f_g + \tilde{c} \odot i_g \tag{12}$$

$\odot$ represents the Hadamard product.

The last part of the cell's internal mechanism is the one responsible for the actual output. It is composed of another gating mechanism, the *output gate*, that decides which one of the cell state values should be passed as outputs after a *tanh* activation.

$$o_g = \sigma(\boldsymbol{W}_{io} \cdot x_i + \boldsymbol{b}_{io} + \boldsymbol{W}_{ho} \cdot h_{i-1} + \boldsymbol{b}_{ho}) \tag{13}$$

$$h_t = \tanh(c_t) \cdot o_g \tag{14}$$

$\boldsymbol{W}_{io}$ and $\boldsymbol{W}_{ho}$ are the matrices storing the output gate projection weights for input and hidden state, respectively; $\boldsymbol{b}_{io}$ and $\boldsymbol{b}_{ho}$ are the tensors storing the output gate biases for input and hidden state, respectively.

The LSTM (Hochreiter and Schmidhuber, 1997) structure has many more variables to optimize with respect to the GRU (Cho et al., 2014), and for this reason, it is not used often in research because it is harder to train. In this case, however, its ability to efficiently model both *long-term* and *short-term* time dependencies was used, thus allowing the learning of complex patterns even using shorter sequences. This led to faster computations and smaller memory usage during training.

2.2.2. Encoder sub-block: CfC-cell

The hidden states computed by the LSTM block at this point still do not account for any time dependency, and thus they are unable to address the unevenness of the passed data. In this context, the CfC layer is employed to help model the time dependencies, adding more robustness to the model. The simplified scheme of the CfC structure is shown in Fig. 6. The structure resembles a RNN Cell, but in this case the gating mechanism serves as an approximation of an ODE equation. A complete and detailed discussion on the topic can be found in the original paper (Hasani et al., 2022). The mathematical operation computed in the cell can be expressed using this equation:

$$h_t = \sigma(-f(\tilde{h}_t, x_t) \odot \Delta t) \odot g(\tilde{h}_t, x_t) + [1 - \sigma(-f(\tilde{h}_t, x_t) \odot \Delta t)] \odot q(\tilde{h}_t, x_t) \tag{15}$$

Where f, g and q are linear operations that project the information from the R^{n+m} space to the R^{n}, meanwhile the two gating mechanisms allow an interpolation between $t \to +\infty$ and $t \to -\infty$. Here $\tilde{h}_t$ represents the LSTM hidden state, x_t is the original input sequence; both are concatenated to compute the new state h_t. This new h_t accounts for the time distance between the input sequence $t-1$ and t, and is the key to the robustness against data sparseness in the proposed framework.

2.2.3. Encoder sub-block: CfC-layer

In the proposed framework, each CfC Layer is realized with the aid of NCP wiring policy (Lechner et al., 2020), so it is composed of three stacked CfC Cells. The peculiarity of this structure is that

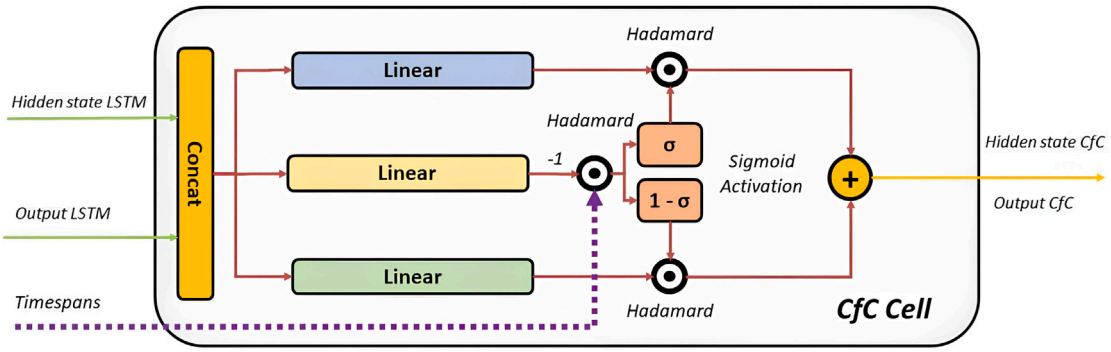


Fig. 6. Schematic representation of the CfC Cell. Even if the information flows similarly as shown in Fig. 1a, this is not an ordinary RNN Cell. The two sigmoid gates perform an ODE approximation, giving h_t the ability to vary accordingly depending on Δt values. In this Cell the three Linear parallel layers approximate the value of the functions f,g,h (Hasani et al., 2022), allowing a fast computation of the ODE approximation. In the figure, *Linear* represents a linear projection layer, σ represents the Sigmoid activation function, *Tanh* represents the Hyperbolic Tangent activation function and $\odot$ represents the Hadamard product.

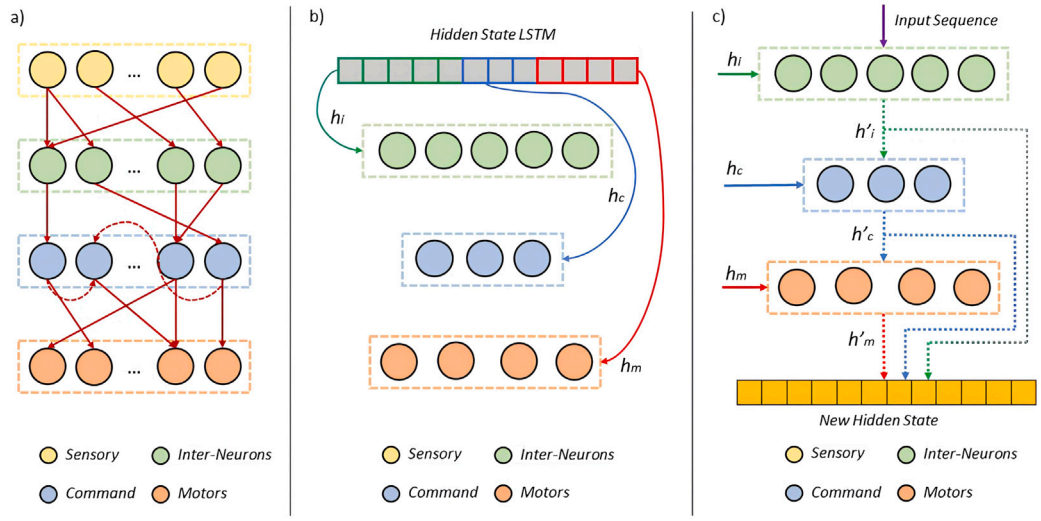


Fig. 7. Schematic representation of the Attention Mechanism used. Attention is performed with a dot product attention, the computations are parallelized in m heads and the results are then concatenated and summed with the original entering tensor.

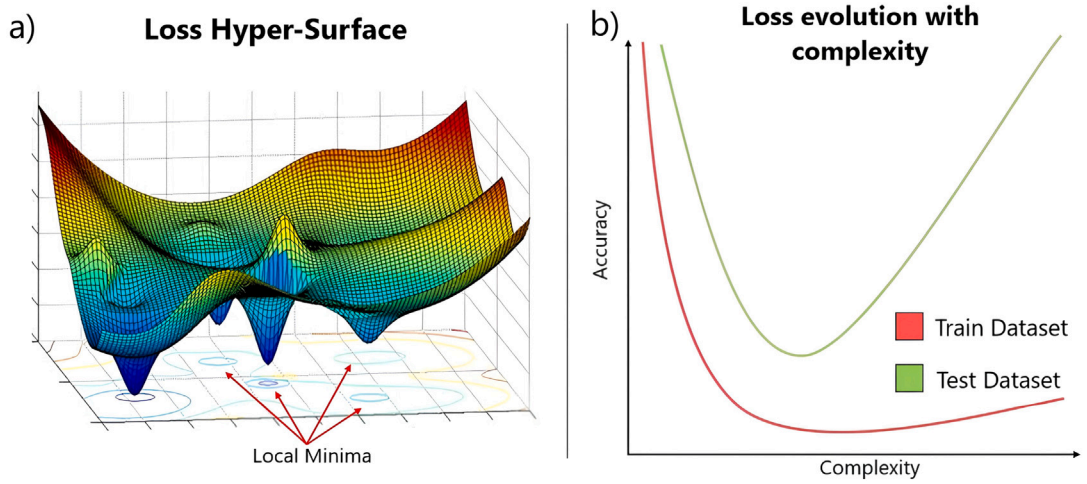


Fig. 8. (a) Proposed wiring based on NCP; (b) The LSTM *hidden state* is split into three tensors that will initialize the initial *hidden state* of three different CfC Layers, wired with NCP; (c) Using the LSTM *hidden states* and the initial *input sequence* new temporal correlated *hidden states* will be computed and concatenated to have the same dimensions of the entering LSTM *hidden states*.

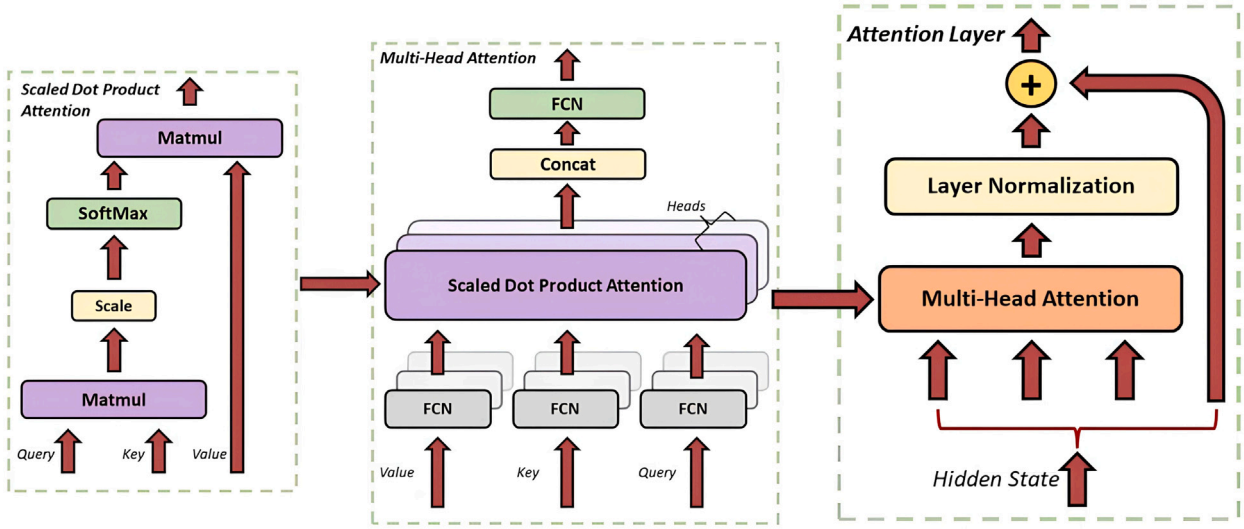


Fig. 9. Qualitative illustration of gradient related issues in deep learning modeling. (a) Loss hyper-surface of an hypothetical loss function. The presence of multiple local minima is highlighted. (b) Evolution of models' accuracy in function of its accuracy.

the connections between neurons in the stacked cells are not fully connected as in a standard network, but instead sparse. The sparsity coefficient is a hyperparameter that can be tuned during the framework initialization by the user or by an agent. The role of the synaptic sparseness in our work has proven to be crucial in improving the NN accuracy because it allows the network to be wider, thus learning more accurate and complex latent representations, without introducing unnecessary variables that could slow down the training process and in general compromise the predictions of the network (Fig. 9b). The intuition behind its efficiency lies in the similarity with another mechanism widely used in NN optimization, the *dropout*. Dropout significantly improves both accuracy and the risk of over-fitting the train data by stochastically zeroing the synaptic responses of a layer during training. The masking mechanism introduced by dropout layers has two main benefits: it allows a better generalization of the training dataset and it detaches a portion of weight and biases from the gradient descent algorithm at each epochs, thus lowering the risk of an unrecoverable fall in a local minima region (Fig. 9a). The sparsity mechanism used in our and Lechner et al. work (Lechner et al., 2020) are similar, but with two significant differences: firstly the synaptic masking is always the same since initialization, secondly the masking is not removed while performing inference. While a re-initialization of the synaptic-masking at each epoch is absent it does not lower the risk of overfitting; it still maintains a beneficial effect towards the gradient descent algorithm and it encourages the development of a more efficient and wide synaptic layer. Similar effects can be also obtained while using the pruning class in Pytorch, or other similar libraries. While using generic pruning classes it allows to lower the computational burden during both the training and inferencing processes, it does not improve the NN accuracy. The key difference in the NCP wiring lies in how the masking is computed and how it differentiates in every sub-layer. The wiring(Fig. 8a) and layer width proposed are the same as used in Lechner et al. work (Lechner et al., 2020) and are summarized as follows:

- Three CfC layers are initialized by using as inputs the LSTM layer width and the output dimension. The width of each CfC layer is defined as follows:

$$\textbf{First layer}(inter\ neurons\ layer) := 0.6 \cdot (LSTM\ width - output\ dimension) \tag{16}$$

$$\textbf{Second layer}(command\ neurons\ layer) := 0.4 \cdot (LSTM\ width - output\ dimension) \tag{17}$$

$$\textbf{Third layer}(motor\ neurons\ layer) := output\ dimension \tag{18}$$

- After creating and initializing the CfC layers, p randomly chosen synapses (where p is the sparsity coefficient, a hyperparameter that could vary from 0.1 to 0.9) are cut off from each layer by both zeroing their coefficients and excluding them from the Gradient Descent algorithm, thus assigning them a constant value through training.
- The second layer neurons are then randomly connected between each other with a number of synapses calculated with the equation:

$$\textbf{Recurrent Synapses} := 0.8 \cdot (LSTM\ width - output\ dimension) \cdot (1 - p) \tag{19}$$

The LSTM neurons are equal in numbers to the sum of the three CfC layers neurons, according to the rules introduced in the previous paragraph, so when the LSTM hidden state is passed, it is divided into three tensors, one for each cell Fig. 8b. The dimension of each tensor is equal to the dimension of the corresponding CfC layer. Each cell then takes the output of the previous cell and the LSTM split tensor as input, as shown in Fig. 8c. After the computation, the $\tilde{h}_t$ states are retrieved and concatenated to give the final ODE approximate hidden state h_t (Fig. 8c). Thanks to this innovative sparseness mechanism, our neural architecture was able to achieve a better generalization of the train dataset, even when we artificially removed data entries from it.

2.2.4. Encoder sub-block: Self multi-head attention mechanism

After the CfC Layer, the output sequence has encoded all the information previously passed by the input sequence, including time information. However, to further improve the framework accuracy a Self Multi-Head Attention Mechanism (Vaswani et al., 2017) is added. This architecture is capable of weighing and giving importance to the most relevant part of the output sequence. The attention mechanism, by doing so, can better use the information flow obtained in the RNN structure and generally this feature is used to prevent memory issues using long sequences. In this research, even if the sequence length does

not cause this kind of issue, we found that the deployment of such architecture generally improves the overall accuracy. Such behavior could be explained by the improved learning efficiency that focused the learning efforts on important data structures/sequences in the train dataset, similarly to the well known mechanism in language models when any kind of attention is used (DeRose et al., 2021; Niu et al., 2021). The dataset used here is not composed of words but data entries, but they still have positional and contextual importance as if they were. In material science, the mechanical properties exhibited by a material are intimately connected between each other, but the high dimensionality possessed by the input data could undermine the development of a valid constitutive law that correlates them. By using an attention mechanism, the model could focus on the important feature, relationship across the dataset, avoiding unnecessary noise and uncorrelated data. The mechanism used here is based on the Self Attention Mechanism concept (Vaswani et al., 2017), so the input sequence is Query, Value and Key (Q, V, K). The structure of the Layer is shown in Fig. 7. Even though the computational burden of this type of attention is high, its impact is significantly lower on both training time and resource utilization when compared to that of LSTM and CfC blocks; thus, in this work other faster alternatives (Wu et al., 2022; Zhou et al., 2020; Dao et al., 2022; Dao, 2023) have not been used. The mathematical operation computed in each head computation can be expressed using the equation:

$$Attention(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \tag{20}$$

In this equation d_k is the dimension used for the heads, it regularize the attention output and it has been proven effective during training in Vaswani et al. (2017).

The information retrieved from each head is then concatenated and projected using a Linear Layer to give the *contextual information*; this tensor will be summed to the original entering hidden state sequence and this new, much more representative, hidden state will be used in the next Encoder block.

2.3. Framework structure: Decoder

The Decoder part in the proposed framework is very similar to the one used in Li et al. (2023a) work. In the current scenario, however, we proposed a variant that uses two parallel FCN to output, at the same time, the target's Mean and Variance. This is particularly useful when the data are scarce, noisy, or with multiple outputs for the same inputs, where the framework can output uncertainty for the prediction. This could inform us whether the model performs well or not, in the interpolation or extrapolation domain. We can further leverage this important feature to perform *active learning* for experiment planning or efficient data sampling. The full framework structure is shown in Fig. 10.

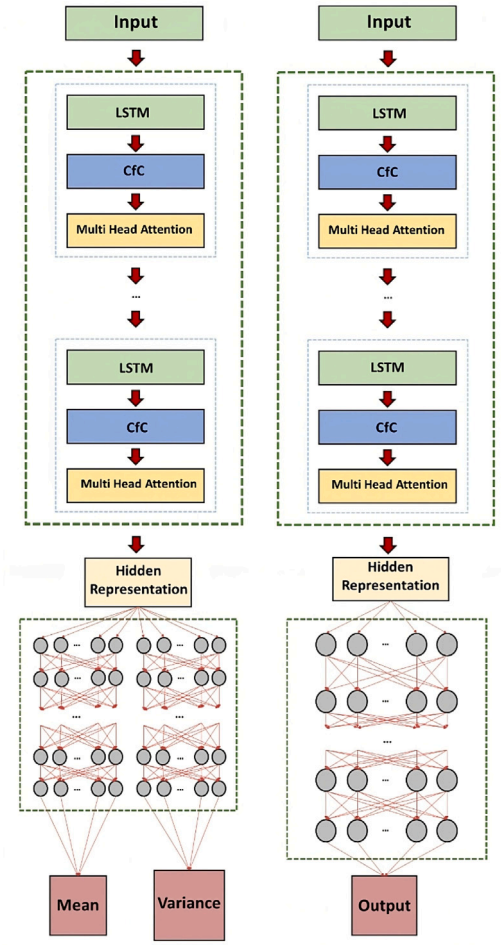


Fig. 10. Full Representation of the proposed Framework. (a) In this variant of the framework, the information after being encoded in a R^p latent space, is decoded using two parallel Fully Connected Network built with the same structure. This approach is particularly useful when the data are redundant, because the learned representation is less susceptible to small input variation and can learn a better generalization of the observed phenomena (Iba, 2020). (b) This is the main structure of the framework. It was mainly used in this research. For its simplicity, if the input data are not noisy or redundant or in general if they do not pose any critical issues, it was often preferred against the Variational one.

2.4. Datasets and their implementation

2.4.1. Datasets

Six datasets were used to test the model performances. The data shown in the next sections increases progressively in dimensionality, data sparseness and noise, thus proving the framework robustness against a large variety of hard to model scenarios. All the dataset that have been used in this research, except the first one kindly provided by Li et al. (2023a), are currently available online to the public.

- The **first dataset** was obtained from uniaxial tensile tests of aluminum sheet samples (Li et al., 2023a). The specimen design and experimental setup are shown in Fig. 11a and in Fig. 11b. The test procedure followed the American Society for Testing and Materials (ASTM) E8/8M-21 standard (ASTM, 2022) for tensile testing of metallic materials. The material testing was performed by both monotonically increasing the load and multiple loading–unloading tests, as shown in Fig. 11c and Fig. 11d. The data used in this study refer only to the multiple loading–unloading scenarios, since the task was more difficult and harder to learn.
- The **second dataset** was obtained from several uniaxial strain-stress curves of AISI 316L stainless steel at different temperatures and strain rates. All the curves used were synthetically obtained by using the Hooke's Law to model the elastic behavior of the deformation and the Johnson and Cook's Law coefficients obtained by Umbrello et al. (Umbrello et al., 2007) to model its plastic behavior. Since elastic rigidity also varies with temperature, the temperature and elastic constant of the material were correlated using a simple Fully Connected Network, to be able to generate *better elastic constant* estimates. A visual representation of the temperature and strain-rate values used is shown in Fig. A.1. The entry points are randomly distributed, which differs from previous work by Li et al. (2023a) and was designed to add further complexity to the learning task.
- The **third dataset** was obtained from uniaxial tensile tests of Al-6061 (Weaver et al., 2016) under different testing angles and aging temperatures (Table B.1) (Weaver et al., 2016). In this

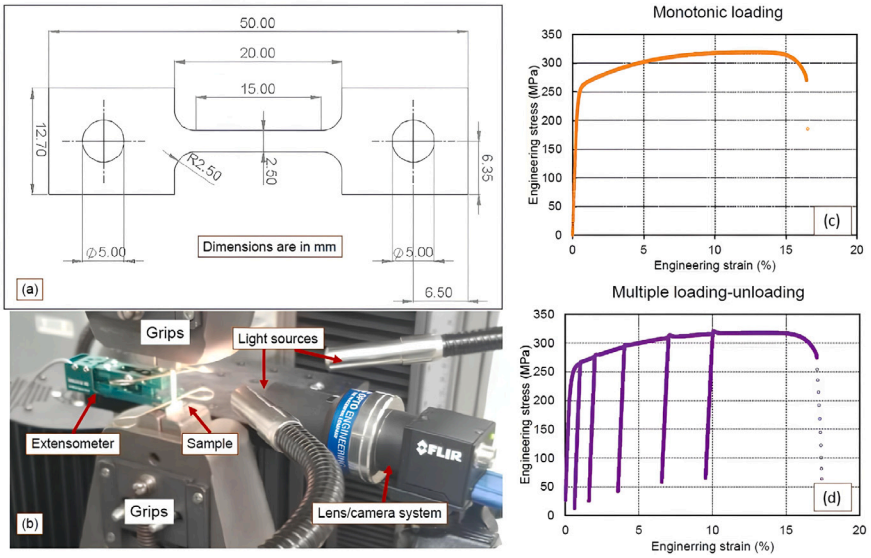


Fig. 11. Experimental tensile tests (a) specimen geometry and (b) experimental setup. Engineering stress and strain curves of (c) monotonically increasing and (d) multiple loading–unloading test specimens. [Taken from Fig. 9 of Li et al. (2023a) with permission].

dataset, a form of anisotropy in the analyzed sample is present, due to the rolling process to manufacture the final specimen. This anisotropy, however, is absent in the aged samples due to the crystal structure relaxation induced by the reheating process. The material mechanical response in this scenario varies significantly, depending on whether the sample is *aged* or *not*. This dataset was used to test the framework's ability to correctly model multiple uncorrelated system dynamics.

- The **fourth dataset** was obtained from uniaxial tensile tests of Al-6061 at different temperatures (Aakash et al., 2019). In this scenario, however, the available grid values were unbalanced (Table B.2), so the learning task was more difficult. Furthermore, the compositions of the samples (Table B.3) were available, so this dataset was also used to further test the framework's robustness against a high-dimensional dataset.
- The **fifth dataset** was obtained from uniaxial tensile tests of laser 3D-printed Inconel 625 at different angles (0°, 30°, 45°, 60°, 90°) and they showed high variance between each other due the high anisotropy of the printing process Benzing et al. (2022). This dataset was used to develop the Variance approach to efficiently tackle the multiple-entries scenario where high variance exists in multiple mechanical tests for material printed at the same conditions.
- The **sixth dataset** was obtained from uniaxial compression tests of Steel spring grade at different temperatures and strain rates (Vode et al., 2019). The entries in this case were much sparser grid-wise (Fig. A.2) and each curve was characterized by a very high noise, so a Savitzky–Golay filter (Press and Teukolsky, 1990) was necessary to smooth its entries. This dataset was used to test the framework's error handling when trained on scarce and highly noisy datasets.

All the data used were artificially damaged since usually strain-stress tests are performed under controlled and constant sample time. To artificially induce a time sparseness inside the data, a uniform random function $\mathfrak{R}$ (0,1) was used to sample, *without replacement* from the original dataset, a number of observations smaller than the original one (Fig. 12a). In a typical used dataset, the observations were more than eight thousand points per stress–strain curve, from which one thousand entries were taken to build the damaged dataset. Some exceptions are present however, but in those cases, the data were sparse and noisy to begin with, so artificial damage was not necessary (e.g., the sixth dataset). In every dataset the loading history contained 100 observations and a zero-padding was added for the first zero strain (Fig. 12b). The zero padding, or padding in general, is not a standard while working with time series because it can easily insert dangerous biases inside the dataset. In this research, the above-mentioned procedure has been used since every set of observations starts with a point characterized by $\varepsilon = 0$ and $\sigma = 0$; thus, using padding composed of zero values does not introduce any biases in the learning procedure. Its usage, moreover, allows the exploitation of the first part of the sequence that would be otherwise lost and not learned.

2.4.2. Train/test/validation splits

Each dataset was split into three groups of *train, test, validation* subsets. The boundary values from the dataset were chosen to test the model accuracy on the extrapolation region and to validate the model performances. Since most of the time the entry data were redundant or, as in the sixth dataset, data points were too scarce to survive further pauperization, artificial entries were created; their performances were evaluated by comparing the output with neighbors' ground truth. The train–test–validation sets were obtained following the criteria described above, leading to the following subsets:

- **First dataset**: the training subset was obtained by performing a random split with a ratio 2:1:1 on 3000 unevenly spaced entries.
- **Second dataset**: the training subset was obtained by generating 100 random curves inside the 30 °C–400 °C and 10^{-3} s^{-1}–10^{5} s^{-1} intervals; the test subset was obtained by using the 4 corners of the training domain; the validation subset was obtained by generating 49 random curves inside the 20 °C–450 °C and 10^{-4} s^{-1}–10^{6} s^{-1} interval, but out of the training domain. As shown in Fig. A.1 the train domain is smaller than the validation one, this has been made to evaluate the model behavior in an extrapolation region far away from the train subset, thus avoiding data-memorization problems. This dataset has been generated using Johnson and Cook's coefficients found in Umbrello et al. (2007). The Temperature and Strain rate points have been chosen by sampling without replacement inside a grid of defined values (inside the validity intervals of Umbrello et al. (2007)), thus testing the framework in a scenario where both spatial and temporal symmetry were absent. In this dataset the ratio between train:test:valid partitions

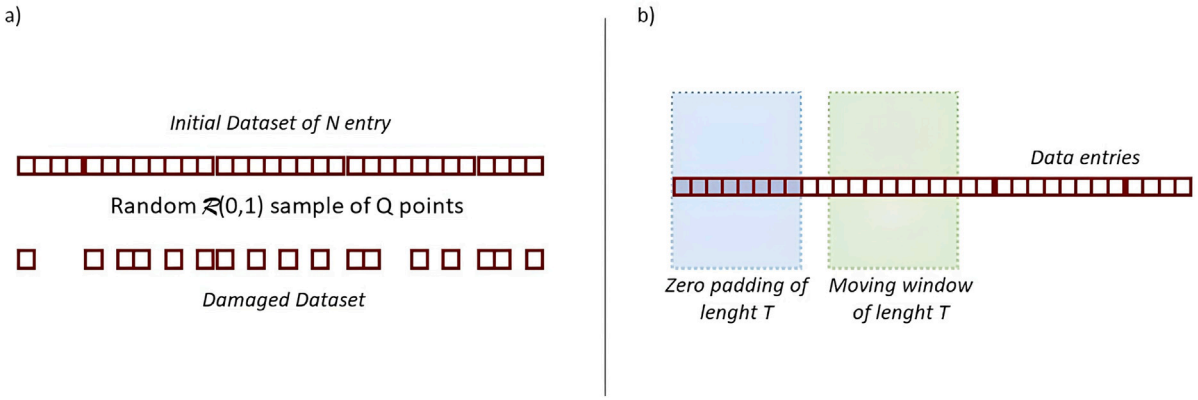


Fig. 12. (a) Damaging procedure. In this research to test the network robustness against a broken time-symmetry, every dataset has been damaged by sampling using a uniform distribution and without replacement N observations (normally N is $[\frac{1}{5}, \frac{1}{10}]$ of the original dataset length); (b) Zero padding mechanism to generate better input sequences.

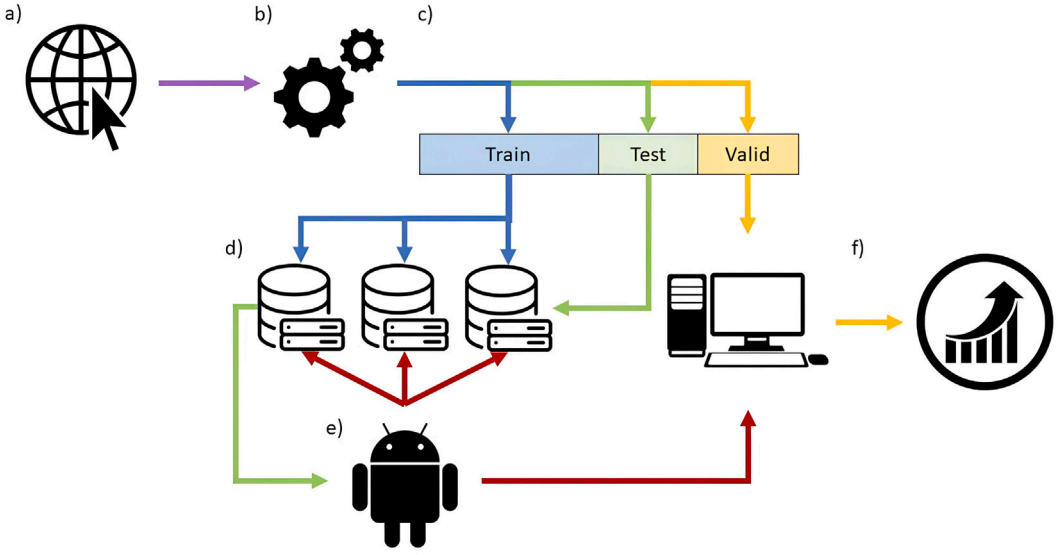


Fig. 13. Schematic representation of the pipeline for the data-retrieving, pre-processing, model training and performance evaluation of each dataset. (a) Each dataset is searched and downloaded from a publicly available repository; (b) Retrieved data is pre-processed, filtered if necessary, and re-scaled in the [0,1] interval to help the Gradient Descend Function to converge; (c) Processed data is divided into Train–Test–Validation subsets when possible. In case of scarcity only Train–Test sub-datasets were created from the original data, meanwhile the Validation was synthesized artificially by using the same data structure of the original dataset; (d) Multiple parallel runs were launched on several remote computers and their Test-Performances sent to an external Agent (e) which based on that, decided the new hyperparameters set for the next runs; (f) After a user termination criterion (normally time or Test Performance), the model with the best Test-Performance was loaded with the Validation subset and the real performance was evaluated.

is 29:1:3. Each curve was composed of 1000 entries, unevenly spaced for training and testing and equally spaced for validation.

- **Third dataset**: the train–test subsets were obtained by using the **Aged** samples at 204.4 °C, 413 °C and the **Not Aged** samples at 0°, 45° and 90° angles as test subset, the remaining as training subset with a ratio train:test:valid equal to 5.7:1:1.8. Due to the redundancy of the entries, a standard validation subset was not obtained by split to avoid biased evaluations; instead, synthetic entries at 375 °C, 315 °C, 250 °C and 230 °C were generated to evaluate the performances on the **Aged** scenario, and synthetic entries at 15°, 30°, 60° and 75° to evaluate the performances on the **Not Aged** scenario. Due to the scarcity of entries, data augmentation using a simple linear interpolation between each couple of points was applied and a random sampling of 1000 entries was performed. The validation subset, being artificial, was generated using:

$$\Delta\varepsilon_{\text{mean}} = 4.37 \times 10^{-5} \qquad \Delta\varepsilon_{\text{max}} = 7.23 \times 10^{-2} \qquad \dot{\varepsilon} = 1.31 \times 10^{-5} \tag{21}$$

Here no damaging procedure was performed, so the validation subset has evenly distributed entries.

- **Fourth dataset**: the train-test subsets were obtained by using the 200 °C Lot A, 100 °C Lot D, 150 °C Lot G, 300° Lot I curves as test subset and the rest as training subset; due to the redundancy of the entries in some regions and the scarcity in others, validation subset was not obtained by split to avoid biased evaluations and generally worse performance; instead synthetic entries were generated at 10 °C, 50 °C, 75 °C, 125 °C, 175 °C, 225 °C, 275 °C, 310 °C, 330 °C with a random composition at the boundaries of the dataset used (Table B.3). In this dataset the ratio between

train:test:valid partitions is 3.8:1:1. The damaging procedure was performed on Train and Test subsets using a random sampling of 1000 entries. The validation subset was generated with $\Delta\epsilon_{\text{mean}}$, ϵ_{max} and $\dot{\epsilon}$ different for each temperature, their values are jointly reported in Appendix B with their random composition values. Here no damaging procedure was performed, so the validation subset has evenly distributed entries.

- **Fifth dataset**: the train-test subsets were obtained by using one curve at each angle (0°, 30°, 45°, 60°, 90°) as the test subset and the rest as the training subset; due to the **high redundancy** of the entries a standard validation subset was not obtained by split to avoid biased evaluations, instead synthetic entries at 0°, 7°, 15°, 30°, 45°, 60°, 75°, 82°, 90° were generated. In this dataset the ratio between train:test:valid partitions is 10:1:1.4. The damaging procedure was performed on Train and Test subsets using a random sampling of 1000 entries. The validation subset was generated by using:

$$\dot{\epsilon} = 4.36 \times 10^{-5} \text{ until } \epsilon = 3.5 \times 10^{-3} \text{ to have 162 data points.} \quad (22)$$

$$\dot{\epsilon} = 2.53 \times 10^{-5} \text{ until } \epsilon = 0.17 \text{ to have 838 data points.} \quad (23)$$

  Here no damaging procedure was performed, so the validation subset has evenly distributed entries.

- **Sixth dataset**: the train-test subsets were obtained by using the 4 corners of the domain as test curves (Fig. A.2), the rest as training subset; due to the scarcity of the entries, the validation subset was not obtained by split to avoid worsening the model performances, instead curves at 1025 °C $10^{-1.5}$ s^{-1}, 1175 °C $10^{-1.5}$ s^{-1}, 1025 °C $10^{0.5}$ s^{-1}, 1175 °C $10^{0.5}$ s^{-1} were generated. Due to the scarcity of entries, data augmentation by using a simple linear interpolation between each couple of points was applied and after that a random sampling of 1000 entries was performed. The validation subset, being artificial, was generated by using:

$$\Delta\epsilon_{\text{mean}} = 2.28 \times 10^{-4} \qquad \epsilon_{\text{max}} = 0.35 \quad (24)$$

  In this dataset the ratio between train:test:valid partitions is 3.9:1:1. Here no damaging procedure was performed, so the validation subset has evenly distributed entries.

2.4.3. Training procedure

To ensure a better convergence and an easier and more effective training a rescaling in the [0,1] interval was performed. In case of logarithmic distributed data such as in the strain rate case, in which the values vary from 10^{-4} to 10^{6}, they were further rescaled by using a base ten logarithmic function before the normal rescaling procedure. To implement the proposed framework Python (Python, 2023a) and Pytorch (Pytorch, 2023b) were used. Due to the large number of hyperparameters involved in the initialization of this framework, a Python library named WandB (Wandb, 2023c) was used to better track and manage their value by using a Bayesian optimizing agent. The operation pipeline is shown in Fig. 13. For each dataset at least 2000 runs have been performed, to guarantee the convergence on the hyperparameters optimization. Unless otherwise stated, the models performed 500 epochs each. To test the performance of the two architectures proposed, the same training procedure described has been applied to both Encoder–Decoder and Variational Encoder–Decoder by using the same pre-processed data. The actual hyperparameter values used to generate the results shown in this research can be found in Appendix C.

3. Results

3.1. Performance of dataset 1: analysis on unseen loading path

The results of the first dataset follow the same criteria presented in the work by Li et al. (2023a), i.e., validating the model performances on the validation subset and by testing further its capabilities on unseen loading scenarios with uniform and random data distribution. The results of both Variational and Normal Encoder–Decoder are shown in Fig. A.4, Fig. A.5 (Normal Encoder Decoder), Fig. 14, Fig. 16 (Variational Encoder Decoder). In this task, the sparseness in time was particularly abundant, as the 14,000 original entries were reduced to 3000 after being damaged and they were further reduced to 1000 after a random train–test–validation split. This process introduces a wide variety in Δt. In this challenging contest, the model convergence was not trivial because there was a very high variance in time and in space between entries. Despite the time and space symmetry being highly broken, the model still demonstrated the ability to predict the material stress–strain behavior in the artificial validation subset.

The predicted behavior in Fig. A.4 and Fig. 14 is very similar to the ground truth, the Variational approach was also able to predict high uncertainty at the end of the resuming loading path (Fig. 14b), thus indicating that such unload–reload transition behavior is not easily predictable and requires further training data in this region. In those areas, both ED and VED networks were unable to successfully predict the transient spikes between elastic and plastic behaviors, in Fig. 14b it can be seen how the Confidence Region (faint red area) has higher values in this regions. In those areas the network fails to accurately predict the transient peaks between the two regions but, at the same time, it is capable of *being conscious* about its errors, giving high value for the output variance in those areas; thus extending the model forecast and signaling the inaccuracy of the prediction.

It should be noted that despite different data structures in the synthetic validation/test set, the model was still able to correctly predict the deformation behavior, which is not a trivial result. Normally when data is fed to the model, if it has a different distribution compared to the training subset, e.g., if the point density is different, predictions could deteriorate. In this case, even though the synthetic data is denser and evenly distributed in the generated path, which was not seen by the model during training, the obtained predictions were valid, suggesting the robustness of the proposed framework.

Uneven highly sparse data was used as input to further prove its robustness, and the results are shown in Fig. A.5 and in Fig. 16. The prediction here shows similar behavior to the one made by the Normal Encoder–Decoder network, but it also shows the *uncertainty* estimation capability previously highlighted in Fig. 14b. It can be seen here how the areas in which the confidence region is higher are the ones characterized by bigger error; thus further demonstrating the network *error handling* capabilities.

Despite the time–space symmetry being highly broken and different from the training dataset, both models were able to predict both mono-loading and multi-loading scenarios similarly to the evenly distributed scenarios (Figs. 14 and A.4). The predicted behaviors are noisier compared to the evenly distributed scenarios, but the general shape of the curves is intact, thus further indicating the robustness against data sparsity. Variational approach was also able, in this challenging task, to output high variance in areas where the prediction accuracy was deteriorating (Figs. 16a and 16b), indicating a promising self-analysis ability that can help to better evaluate predictions in challenging scenarios.

Comparative results have also been generated by using two optimized NNs with an Encoder composed by 3 GRU layers (Fig. 15b) and by 3 LSTM layers (Fig. 15a) respectively. Both networks were unable to efficiently learn the material behavior by using the training data, as it is shown in Figs. A.6a, A.6b, A.7a, A.7b. The GRU Net shows a reduction in accuracy on the elastic path when uniform data is used; the loading and unloading paths predicted in Fig. A.6b are too wide and unrealistic when compared to the experimental ground truth (A.6a, Fig. A.6b). These findings were expected, because the time-symmetry was absent in the training dataset, thus the network was unable to efficiently time-correlate its *hidden states*. The network behavior worsens when an uneven data distribution is used, in Fig. A.6c, Fig. A.6d the network

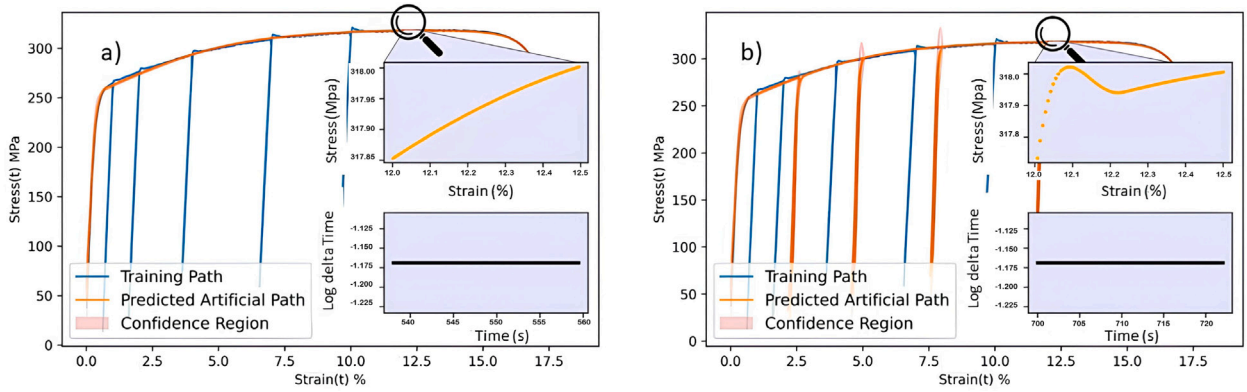


Fig. 14. Variational Encoder–Decoder predictions on the first dataset. (a) VED Prediction on the synthetic unseen mono-load path; (b) VED Prediction on the synthetic unseen multi-load path. Here both mono and multi-loading paths have a different distribution (as it can be seen in the purple highlighted sub-plot), from the training dataset, similarly at what it has been shown in Fig. A.4.

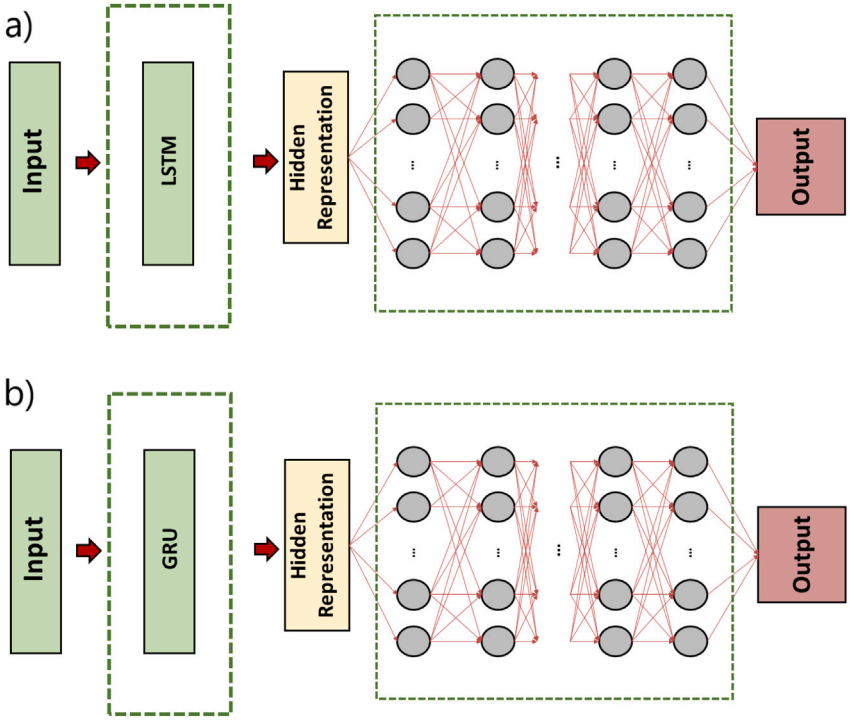


Fig. 15. Comparative RNN architectures. (a) LSTM model architecture; (a) GRU model architecture. The difference between them and the architecture proposed in this work lies on the absence of both MHAL and CfC layers.

fails to model the material elastic behavior, and the overall predictions are very noisy. Interestingly the LSTM Net performed worse than the GRU one (Figs. A.7a, A.7b) and this could be caused by its more complex and hard to optimize structure. In (Fig. A.7c, Fig. A.7d) this effect is more evident, because the data distribution, was even less uniform than the training one. The predictions here are noisy, and the model fails to accurately predict both the material elastic and plastic behavior, thus indicating the CfC Layers as the main contributors for the accuracy showed by the models used in this work. Since the standard RNN architectures were unable to model one of the easiest time-independent data structures, it is proofed their inability to model such scenarios. As such, comparative results will not be shown for the other five datasets.

3.2. Performance of dataset 2: analysis in extrapolation region

For dataset 2 task, the model was tested by exposing it to unseen test conditions, namely: **Temperature** and **Strain Rate**. To faithfully test the extrapolation capabilities, the validation domain was chosen to be out of the training domain. 49 validation curves were used for evaluation (the extended results can be found in the GitHub repository). Here we only show the 4 areas near the corners of the validation domain 20 °C 10^{-4} s^{-1}, 20 °C 10^{6} s^{-1}, 450 °C 10^{-4} s^{-1}, 450 °C 10^{6} s^{-1} (Fig. A.1). As it is shown in A.1, the artificial dataset used for the task, is composed of sparse and random temperature and strain rate entries, so some areas are characterized by a high level of sparseness,

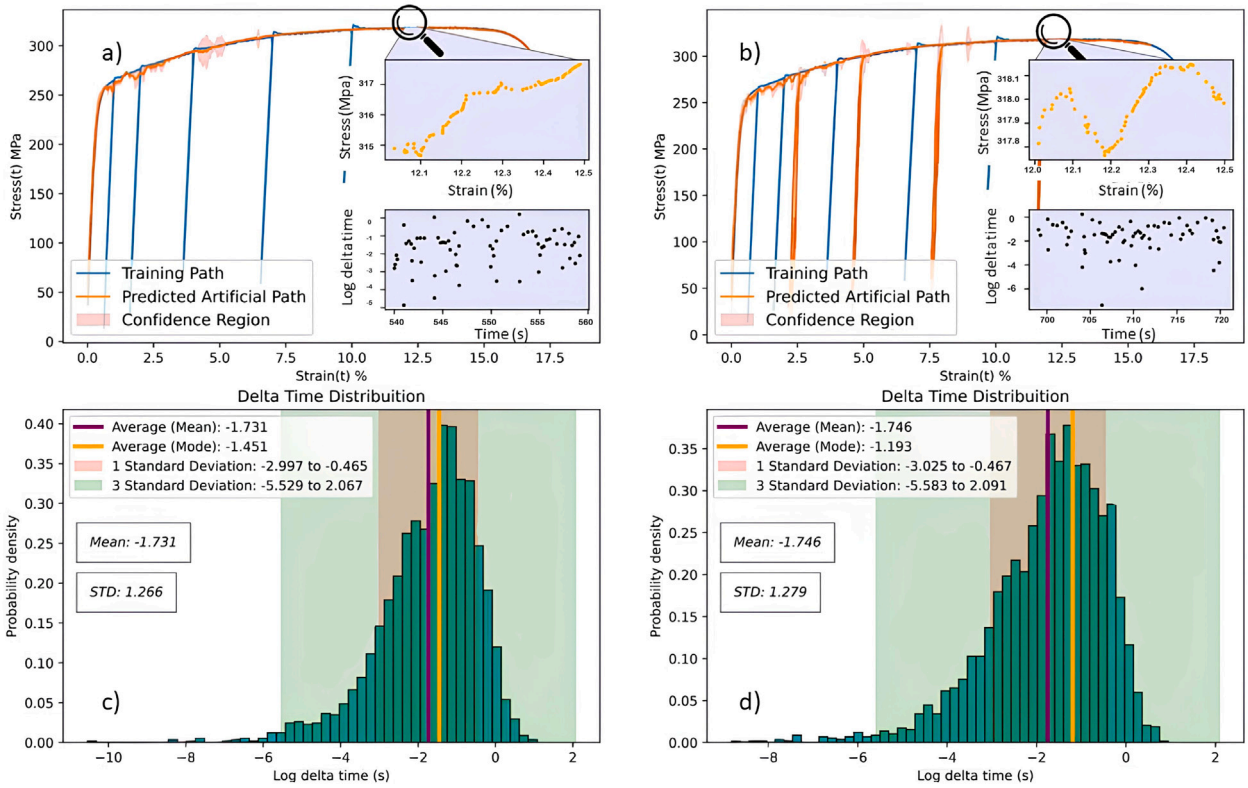


Fig. 16. Variational Encoder–Decoder predictions on the first dataset. (a) VED Prediction on synthetic unseen not uniform mono-load path; (b) VED Prediction on synthetic unseen not uniform multi-load path; (c) Mono-load Δt distribution; (d) Multi-load Δt distribution. Here both mono and multi-loading paths have a different distribution (as it can be seen in the purple highlighted sub-plot and in the sub-figures *(c)* and *(d)*), from both training dataset and Fig. 14 validation paths. They have a Δt characterized by a very high variance, similarly at Fig. A.5; in those validation paths Δt spans from 10 s to 10^{-10}s.

such as in the right lower corner near 20 °C 10^{6} s^{-1} and in the left lower corner near 20 °C 10^{-4} s^{-1}. Such large sparseness in training data, together with the fast-changing behavior of the material at low temperatures, led to a severe loss in model accuracy, highlighting the need for more training data in those areas. Even if the framework can handle high levels of sparseness as input, the training dataset must be denser in areas intrinsically difficult to model.

The results are shown in Fig. 17 and refer to the Variational Encoder–Decoder, which performed better in this task. As shown in Fig. 17 the prediction are characterized by a good accuracy at higher temperatures (Figs. 17a and 17b), but by a lower accuracy at lower temperatures (Figs. 17c and 17d). In Fig. 17a at both 400 °C and 450 °C at 10^{-4}, can also be seen a reduction in accuracy at the beginning of the curve compared to the whole high-temperature points cluster. This behavior was expected, because mechanical responses change very quickly with Temperature and Strain Rate and in this particular scenario, both the lower and the left edge of the domain are not densely populated enough due to the grid generation stochasticity (Fig. A.1). While at high temperatures the observations population were denser enough to compensate for the sparseness at lower strain rates, at lower temperatures the sparsity of the observation in both fields caused a sudden accuracy drop. In this case, however, the network accuracy could be easily fixed by populating those dataset regions with more observations. Despite those considerations, the model was able to predict a good mechanical response until 0.15 strain for 20 °C at 10^{-4} and 10^{-3} s^{-1} and 0.25 for 10^{5} and 10^{6} s^{-1}, highlighting model robustness against data sparsity even when used to predict out of domain conditions.

3.3. Performance of dataset 3: analysis on different dynamics

For dataset 3 task, the model was tested on unseen test conditions, namely: **Aging Temperature** and **Angle of Testing**. In this task only the Variational approach gives accurate predictions in the evaluation procedure, therefore we only discuss the results from the variation approach. As we do not have the ground truth behind the prediction, forecasting validity was examined by comparing the predictions and their available neighbors in the original dataset.

The model performance is shown in Fig. 18 and in Fig. A.8. In Fig. 18a, at an aging temperature of 230 °C, the prediction has a shape consistent with the 4 nearest curves. Moreover, the prediction values are also consistent with the trend exhibited by the aged samples. In the proposed dataset, higher aging temperatures are associated with lower stress levels; this trend is visible in Fig. 18. The prediction at 230 °C has values between that of 204 °C and 274 °C curves, which can be considered plausible and coherent with the real-world observations. The same argument can be made for all the other predictions shown in Fig. 18. Fig. A.8 shows the results of the non-aged predictions. In this case, the properties vary in a much smaller domain, but the same argument can be made as before. In this case, the lowest registered mechanical properties were associated with a testing angle of 45°, with a maximum at 90° and a mean value at 0°. The predictions (Fig. A.8) in this scenario are consistent with the shape of the neighbors, with stress values always greater than the minimum at 45° and lower than the maximum at 90°. Being more specific, the prediction at 15° (Fig. A.8a) is close to the mean value at 0°; the prediction at 30° (Fig. A.8b) is close to the minimum boundary at 45°; and the prediction at 60° and 75° (Figs. A.8c and A.8d) are also close to the maximum boundary at 90°. In conclusion, the predicted behavior of the material in the **Non-Aged** scenario is consistent with the ground-truth observations.

Some artifacts are, however, present at the very beginning of the curves. This is due to the absence of data in that region of the training curves. This uncertainty was also captured by the model, e.g., the Confidence Region is wider, so the model itself is aware of the untruthfulness of its predictions in that domain. Such a problem can be, however,

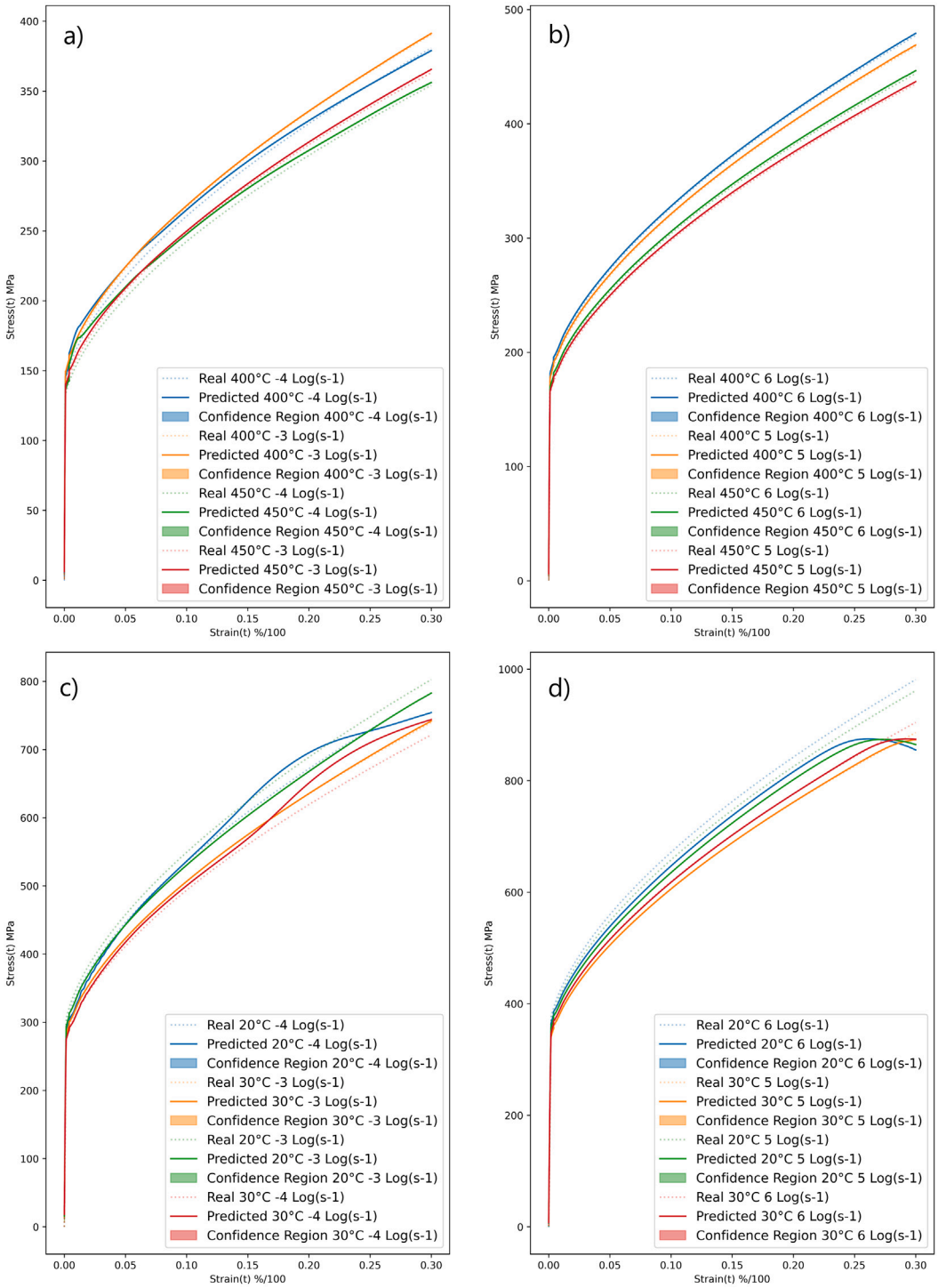


Fig. 17. VED prediction on the corners of evaluation subset. In order in the figure is shown the 450 °C–400 °C and 10^{-4}–10^{-3} s^{-1} corner area (a), the 450 °C–400 °C and 10^{5}–10^{6} s^{-1} corner area (b), the 30 °C–20 °C and 10^{-4}–10^{-3} s^{-1} corner area (c), the 30 °C–20 °C and 10^{5}–10^{6} s^{-1} corner area (d).

easily solved by feeding the network with better data in the affected area. It is worth noting that the Variational Encoder–Decoder was also able to capture some degree of uncertainty near the material yield, due to the high variance of the training sample in that region. This behavior is stronger in the **Non-Aged** scenario, because the variance of the training samples was higher. This is an important result because the model is capable of estimating uncertainty changes in relation to the strain values, Aging Temperature and Testing Angles, fully capturing the complex real system dynamics.

3.4. Performance of dataset 4: analysis on unbalanced dataset and high dimensional inputs

Models for the dataset 4 task are tested on unseen test conditions **Test Temperature** and **Composition**. The temperature in the evaluation dataset was specified in Section 2.4.2, meanwhile the compositions were chosen by randomly sampling in the boundary of the dataset domain. The values of the compositions for each curve can be found in Table B.3. This task was particularly difficult for the framework,

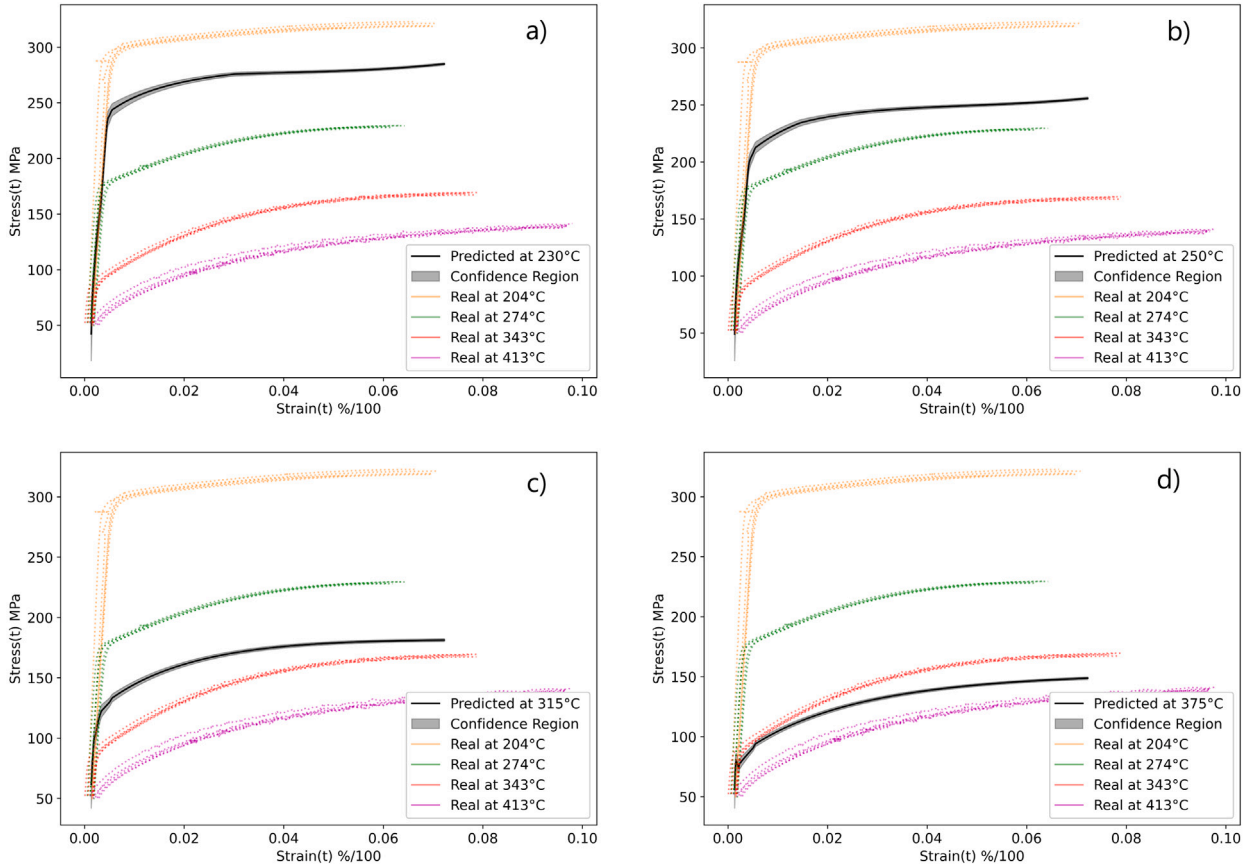


Fig. 18. Variational Encoder–Decoder prediction on the third dataset evaluation subset in Aging conditions. (a) Prediction at 230 °C; (b) Prediction at 250 °C; (c) Prediction at 315 °C; (d) Prediction at 375 °C. Here it can be seen in every sub-plot how the model has successfully modeled the mechanical behavior of the material; each of the predictions has a shape consistent with one of its neighbors and correlated with its Temperature. The Confidence Region also has higher values near the yield zone in each of the predicted curves, accordingly with the variance expressed by the training samples in the same area.

due to the high dimensionality of the input tensor, and an unbalanced dataset with zones characterized by multiple entries with high variance and different curve lengths. The performances are shown in Fig. A.9 and in Fig. 19, for both the Encoder–Decoder framework, and the Variational one. Similar to the dataset 3 task, the ground truth behind the prediction is unavailable, therefore the predictions were examined by comparing them to their available neighbors in the original dataset.

As shown in Figs. 19 and A.9, the overall shape of the predictions is consistent with that of the real curves. Specifically, in Fig. A.9d at the temperature of 125 °C, the predictions are between the real curves at 100 °C and 150°. In Fig. A.9a, at the temperature of 10 °C, the predicted values are bigger than the real ones at 20 °C, suggesting that the framework is able to extrapolate properly. These results further demonstrate that the model predictions closely follow the real system dynamics. We note that the predictions from the normal Encoder–Decoder are more accurate, which probably is due to the smaller number of variables employed and consequently a better convergence during the optimizing process.

The Variational Encoder–Decoder, on the other hand, has less precise predictions compared to that of the Normal Encoder–Decoder; however, the uncertainty handling allows evaluating a Confidence Region around ground-truth observations, in this case, high variances (Fig. A.9). In Fig. 19, the Variational approach has a very large variance compared to the one exhibited in Fig. A.9 thus indicating that the predictions are not correct and thus labeling the domain region as potential sampling region in an active learning process. The compromised framework capabilities may be caused by the very high variance and fewer data points exhibited by the real curves in that domain region. While the Normal Encoder–Decoder performed better than the Variational one, the high variance from the latter is also useful in a practical scenario. For example, a high predicted variance may indicate an untruthful prediction (Fig. 19), or scattered testing/samples quality, etc., which will help the research team to plan for the next experiment. This finding further assess what it has been shown in Section 3.1 in Fig. 14b.

3.5. Performance of dataset 5: analysis on redundant and high variance inputs

We examined the model performance for dataset 5 task, by testing the model on unseen **Angles**, which are specified in Section 2.4.2. This dataset was used to evaluate and develop the Variational Encoder–Decoder in a scenario where numerous entries are available for nominally the same loading conditions. In this context, the data collected refer to Inconel 625 samples obtained by Laser Printing. This process, even if conducted in a controlled environment, generates samples characterized by a certain level of variance caused by the intrinsically high number of random variables. The high value of variance in the samples cannot be then ignored and must be reported and used in future uses and real applications. In this context, having a framework able to model the behavior of the material jointly with its uncertainty in unseen scenarios is crucial.

The model performances are shown in Fig. A.10 and Fig. 20, since this task was intended for a Variational approach, only the results

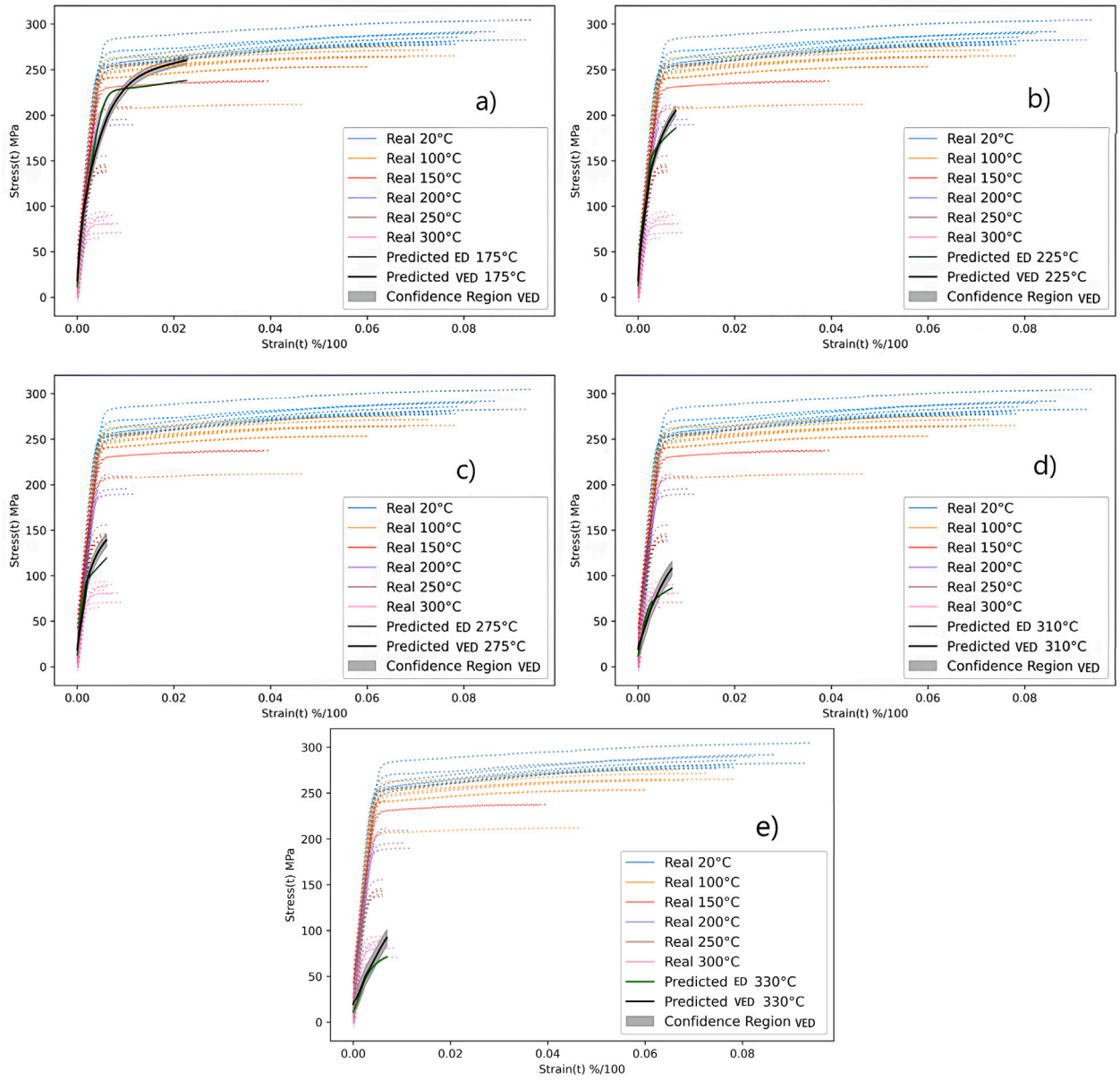


Fig. 19. Variational(black) and Normal (green) Encoder–Decoder prediction on the fourth dataset evaluation subset. (a) Random composition at 175 °C; (b) Random composition at 225 °C; (c) Random composition at 275 °C; (d) Random composition at 310 °C; (e) Random composition at 330 °C. Here the predictions show consistency with their neighbors both in shape and temperature-dependent mechanical behavior, however the VED network shows a significant accuracy loss when compared to the standard ED.

obtained with the Variational Encoder–Decoder are shown. Fig. A.10 shows the performances on the artificial strain path, with angles that the model has already seen during the training procedure. Even though the data distribution is different for inference respect to the training one, the predictions are consistent. The model in this scenario can precisely predict both the mean value and the variance present at different angles.

Fig. 20 shows the model predictions on artificial strain paths at unseen angles. Again, due to the lack of ground truth observations, we did a sanity check on the predictions by comparing them with their available neighbors in the original dataset. It can be seen that the stress level has a maximum at 90°, and it worsens when the angle becomes smaller until it reaches its minimum at 0°. In this context, the model predictions are consistent both in shape and in values because they always follow both the trend and the variance as discussed, suggesting the framework validity in this task.

3.6. Performance of dataset 6: analysis on scarce and very noisy inputs

Finally, we test the models for dataset 6 task on unseen test conditions: **Test Temperature** and **Strain Rate**. See Section 2.4.2. for more details on the evaluation dataset (both temperatures and strain rates). The task associated with this dataset is conceptually very similar to that for dataset 2, but it differs in some crucial aspects:

- Data entries used in this task are intrinsically noisier and dataset 6 represents a very noisy case that requires a Savitzky–Golay

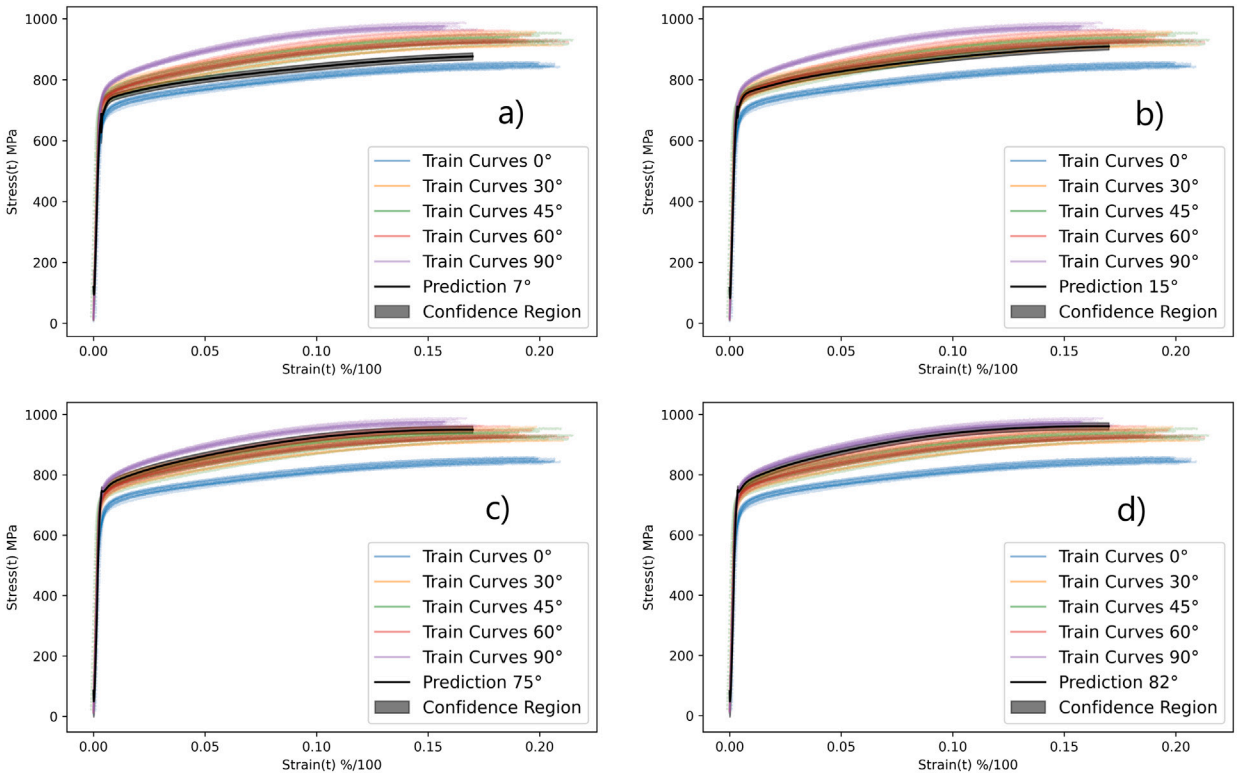


Fig. 20. Variational Encoder–Decoder prediction on the fifth dataset evaluation subset. Only the angle values absent in the real dataset are shown. Real observations are also reported so as to allow practical considerations between the predictions and their neighbors. In order predictions at 7° (a), 15° (b), 75° (c), 82° (d) are reported. In the subplots it can be seen how the network predictions are consistent in shape, trend and variance with the ground truth observations.

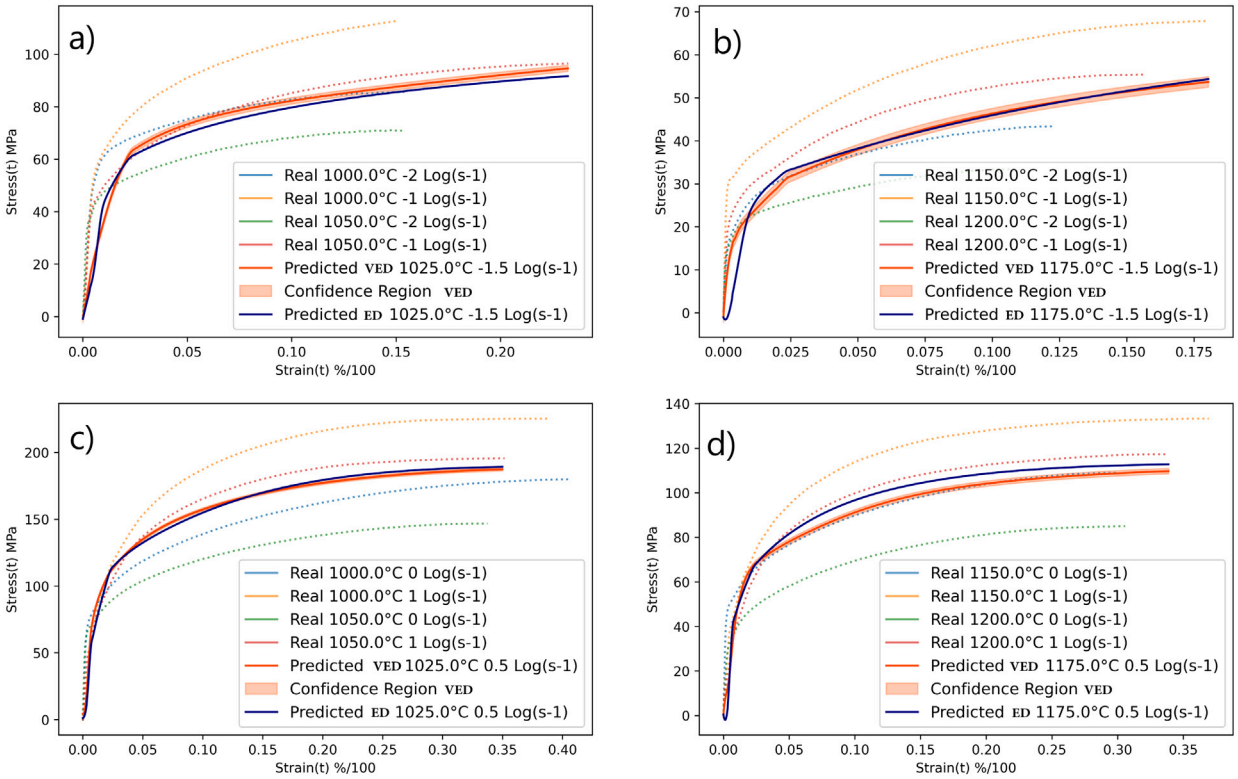


Fig. 21. Variational (orange red) and Normal (dark blue) Encoder–Decoder prediction on the sixth dataset evaluation subset. Being completely artificial, the neighbors ground truths (Fig. A.3) are also shown in the figure to allow a practical evaluation. In order prediction at 1025 °C and $10^{-1.5}$ s^{-1}(a), 1175 °C and $10^{-1.5}$ s^{-1} (b), 1025 °C and $10^{0.5}$ s^{-1} (c), 1175 °C and $10^{0.5}$ s^{-1} (d) are reported.

filter (Press and Teukolsky, 1990) to smooth and eliminate the excessive variance.
- Even if data Grid entries of Temperatures and Strain Rates are regular, they were fewer, making the task more difficult to learn.

Due to the above challenges, the model accuracy for this task is relatively lower compared to the other scenarios. However, the model can still be considered to have good prediction capabilities, as justified by comparing the predictions with their available neighbors (Fig. A.3) in the original dataset.

The results are shown in Fig. 21, for both the standard Encoder–Decoder framework and the Variational one. The predictions made have a consistent shape with their neighbors and the value predicted is consistent with the trend exhibited by the ground truth observations, where for lower temperatures and faster strain rates is noted an increase in mechanical response from the material. In this scenario however, due to both scarceness and noisiness of the data, ED fails to capture the correct elastic behavior of the material. This problem does not affect VED, thus highlighting how the deployment of two heads in the decoder helps the model to understand noisier observations.

The overall performances for both approaches are very similar; however, the Variational framework was able to give a smoother transition at the beginning of the predicted strain-stress curve. In addition, even though the Variational framework has more complexity in architecture, it showed better convergence during training, thus suggesting better usage of the data, especially if the data is noisy, as in this case.

Summary

Based on the previous encoder–decoder approach by Li et al. (2023a), we have proposed a new framework to tackle the unevenness in input data sequence for constitutive relation modeling. The proposed framework utilizes CfC (Hasani et al., 2022) with a new synapse wiring technique NCP (Lechner et al., 2020) to mimic the role of Ordinary Differential Equations, and can make predictions on the mean and variance of stresses. The proposed network has been demonstrated on six different datasets with various complex data structures and noise levels. The results (summarized in Fig. A.11)suggest that our framework is able to model data distributions with uneven input sequences in every dimension, thus further extending NN-based constitutive modeling to the domain with non-structured datasets. Specifically, the framework was trained, validated, and tested using six different strain-stress datasets, each characterized by its own challenges, respectively, multiple loading paths, out of domain inference, highly sparse inputs, high-dimensionality inputs, redundant inputs, and unbalanced and noisy dataset. To further prove its robustness and generalization capability, the same framework structure has been used across every dataset, only varying the hyperparameters of the various layers. The models for every dataset not only showed good training convergence but also demonstrated reasonably good (quantified error estimation is unavailable due to the lack of ground truth) predictions in the extrapolation regions. The capability of being able to predict the mean and variance allows the modeling of source data distribution (such as in 3D printed materials with high variance even in the same batch printed material) as well as uncertainty estimation in prediction itself, which will benefit scenarios with noisy data entries and potential integration into an active learning/Bayesian optimization process. Such a feature could accelerate the materials development/optimization process by informed decision-making that balances exploration and exploitation. Finally, from a practical point of view, the proposed framework demonstrated robustness in dealing with tough data such as heavily damaged, temporally uncorrelated, noisy and uneven datasets, as well as different data distribution in *inference mode*, thus allowing better data usage and increasing the applicability and reliability of NN-based constitutive modeling in real-world applications where a uniform sampling with high signal to noise ratio in both time and space cannot be achieved and where data abundance is absent due to the cost associated with the data acquisition process.

In future research the time-adaptive capabilities of the CfC network could be further explored by combining NCP wiring and Genetic Algorithm (Tomczak, 2022). In the current work both the *fanin* and *fanout* of sensory, inter, command and motor neurons as well as their numbers have been maintained fixed in each sub-block to limit the hyperparameter search space and to allow the Bayesian Optimization Algorithm to converge faster. However, Genetic Algorithms can remove the above restrictions and provide an efficient hyperparameter space navigation, thus allowing faster and better convergence. By using such techniques, the aforementioned wiring information could be encoded within a chromosome for each sub-block separately,thus allowing more flexibility and better performances both in computational time and accuracy. Similar workflows have been successfully applied to automatically searching and building hyper-optimized NNs for Image classification (Loshchilov and Hutter, 2016; Sun et al., 2020, 2019; Bakhshi et al., 2019) or Image Generation (David and Greental, 2014). Furthermore, even CfC cell structure could be improved or completely revolutionized by using Genetic Programming (GP). In recent years, several works (Chapter 9 and 10 of Tomczak (2022)) have successfully used genetic programming in Deep Learning (Galván and Mooney, 2021; Gavrilescu et al., 2022) to develop cells with an unseen gating mechanism and structure capable of outperforming all ever discovered RNNs. These new developments provide new oppurtunities to create NNs even faster and more robust to noise, temporal irregularities, and data distribution.

Code availability

The code used in this paper is under the Apache 2.0 Licence and can be found in the publicly available repository on GitHub at: https://github.com/Marcelaus98/CfC-for-Material-Science.

CRediT authorship contribution statement

Marcello Laurenti: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Data curation. **Qing-Jie Li:** Writing – review & editing, Supervision, Conceptualization. **Ju Li:** Writing – review & editing, Validation, Supervision, Project administration.

Declaration of competing interest

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

Data availability

I have shared both data and code on my Github page. The link was shared in the appropriate section in the paper.

Acknowledgments

We acknowledge support by Eni S.p.A. through the MIT Energy Initiative.

Marcello Laurenti acknowledges support by AIDIC (Associazione Italiana di Ingegneria Chimica).

Appendix A. Auxiliary figures

See Figs. A.1–A.11

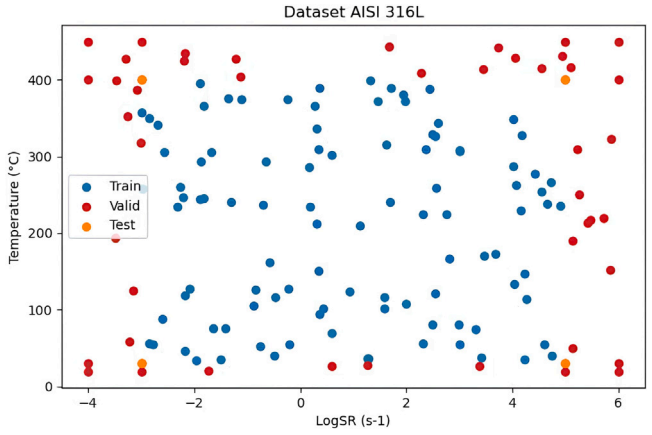


Fig. A.1. Visual representation of the train/test/valid for the second dataset: the horizontal axis shows the logarithm of the strain rate, the vertical axis the temperature in Celsius.

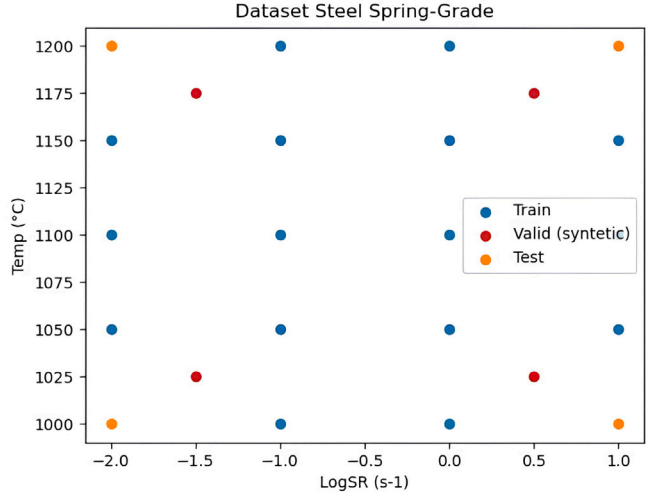


Fig. A.2. Visual representation of the train/test/valid for the sixth dataset. The horizontal axis shows the logarithm of the strain rate, the vertical axis the temperature in Celsius. In this case the dataset was real, so the observation are equally spaced in both Temperature and Strain Rate.

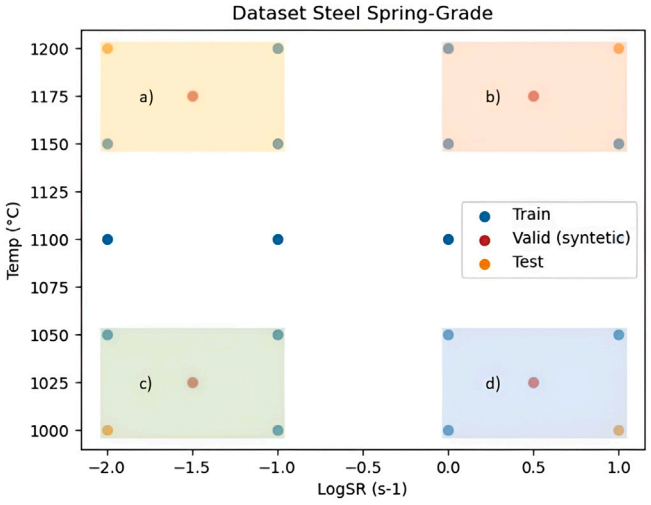


Fig. A.3. Practical representation of the sixth dataset methodology for evaluating validation subset predictions by using the neighbors' ground truths. In order, in the light shaded areas there are the dominia used for the evaluation of the predictions at 1175 °C and $10^{-1.5}$ s^{-1} (a), 1175 °C and $10^{0.5}$ s^{-1} (b), 1025 °C and $10^{-1.5}$ s^{-1} (c), 1025 °C and $10^{0.5}$ s^{-1} (d).

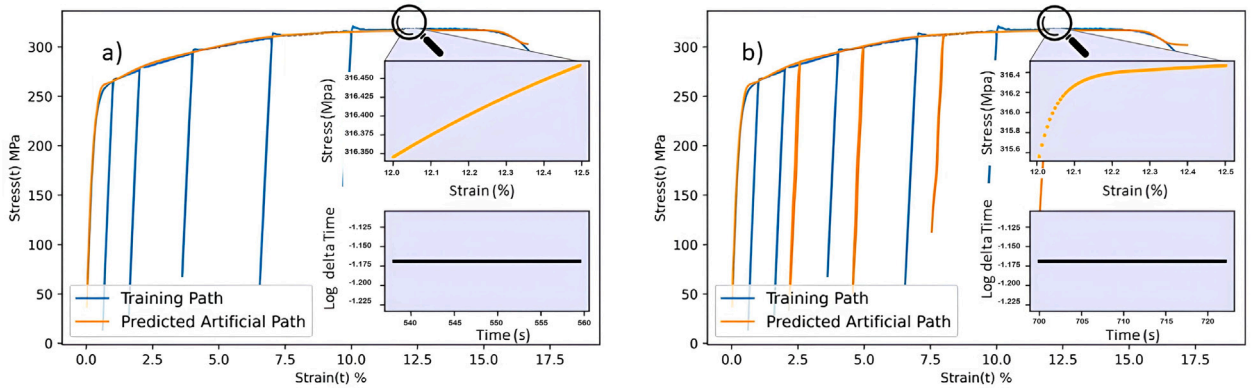


Fig. A.4. Normal Encoder–Decoder predictions on the first dataset. (a) ED Prediction on synthetic unseen uniform mono-load path; (b) ED Prediction on synthetic unseen uniform multi-load path. Here both mono and multi-loading paths have a different distribution (as it can be seen in the purple highlighted sub-plot), from the training dataset. The prediction accuracy while using these validation paths is very high and highlight how, thanks to the CfC cells, the network was able to successfully reconstruct and learn the time dependencies present in the train dataset.

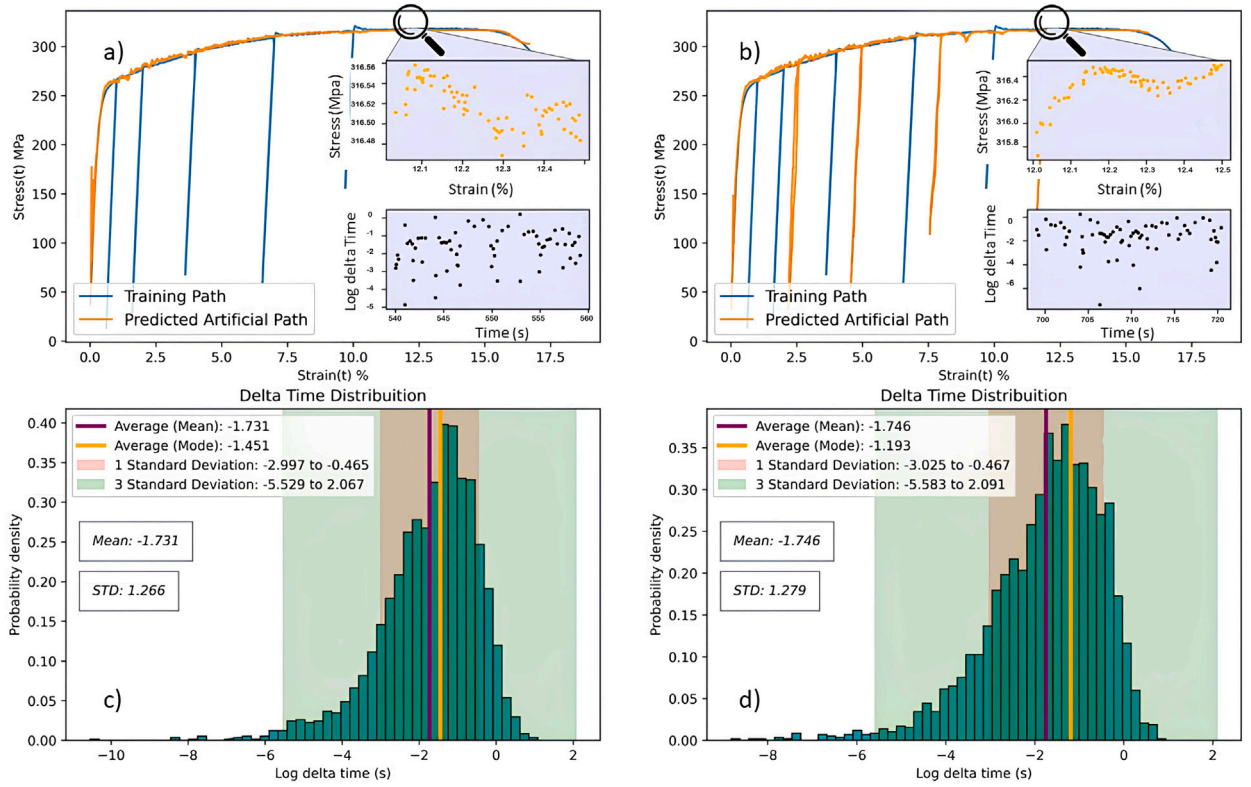


Fig. A.5. Normal Encoder–Decoder predictions on the first dataset. (a) ED Prediction on synthetic unseen not uniform mono-load path; (b) ED Prediction on synthetic unseen not uniform multi-load path; (c) Mono-load Δt distribution; (d) Multi-load Δt distribution. Here both mono and multi-loading paths have a different distribution (as it can be seen in the purple highlighted sub-plot and in the sub-figures *(c)* and *(d)*), from both training dataset and Fig. A.4 validation paths. They have a Δt characterized by a very high variance; in those validation paths Δt spans from 10 s to 10^{-10}s. The prediction accuracy here, while noisier compared to Fig. A.4, is still high, further demonstrating how the network is very robust to extreme unevenness during both *training* and *inference mode*.

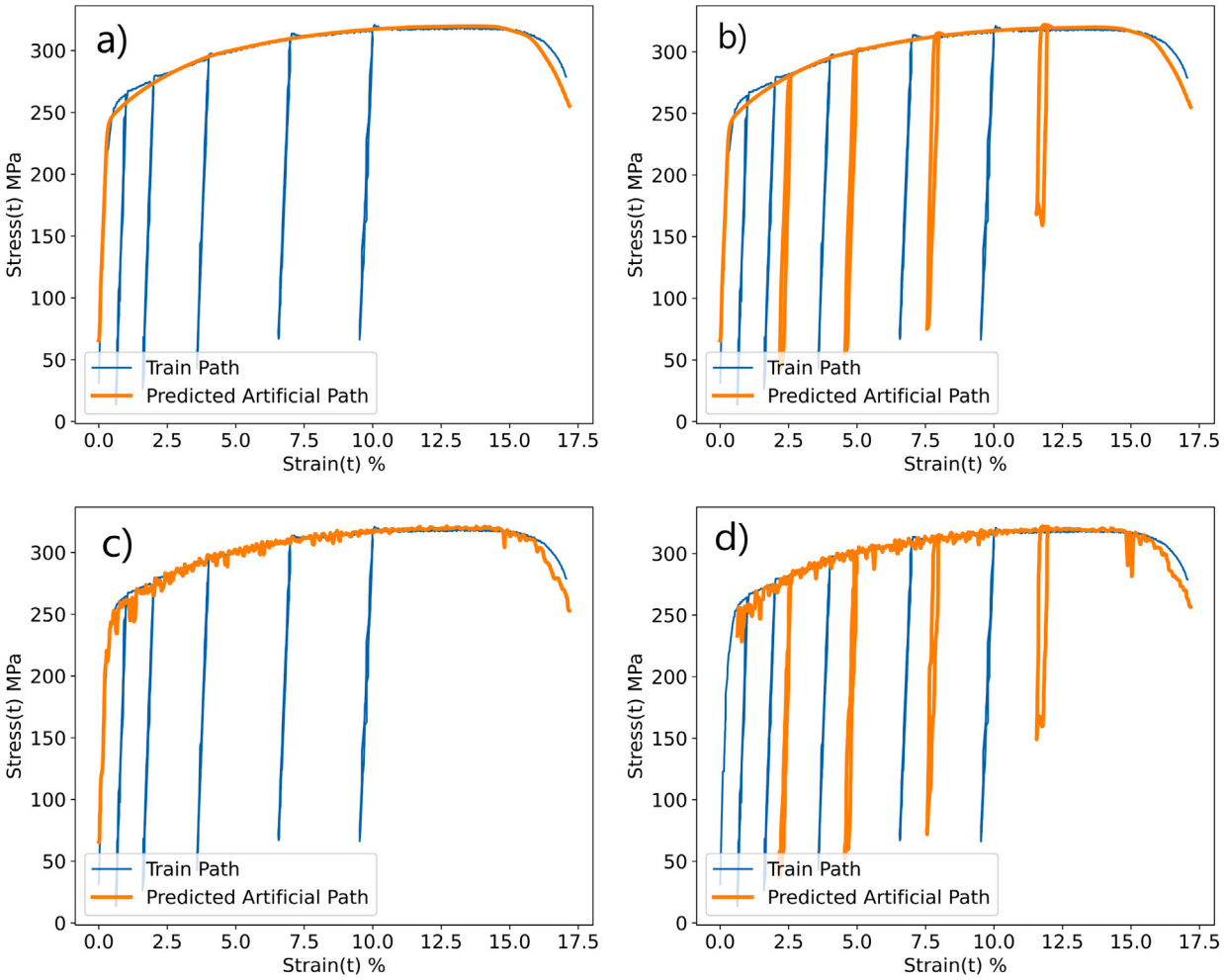


Fig. A.6. (a) GRU Net prediction on synthetic unseen uniform mono-load path; (b) GRU Net Prediction on synthetic unseen uniform multi-load path; (c) GRU Net prediction on synthetic unseen not uniform mono-load path; (d) GRU Net Prediction on synthetic unseen not uniform multi-load path.

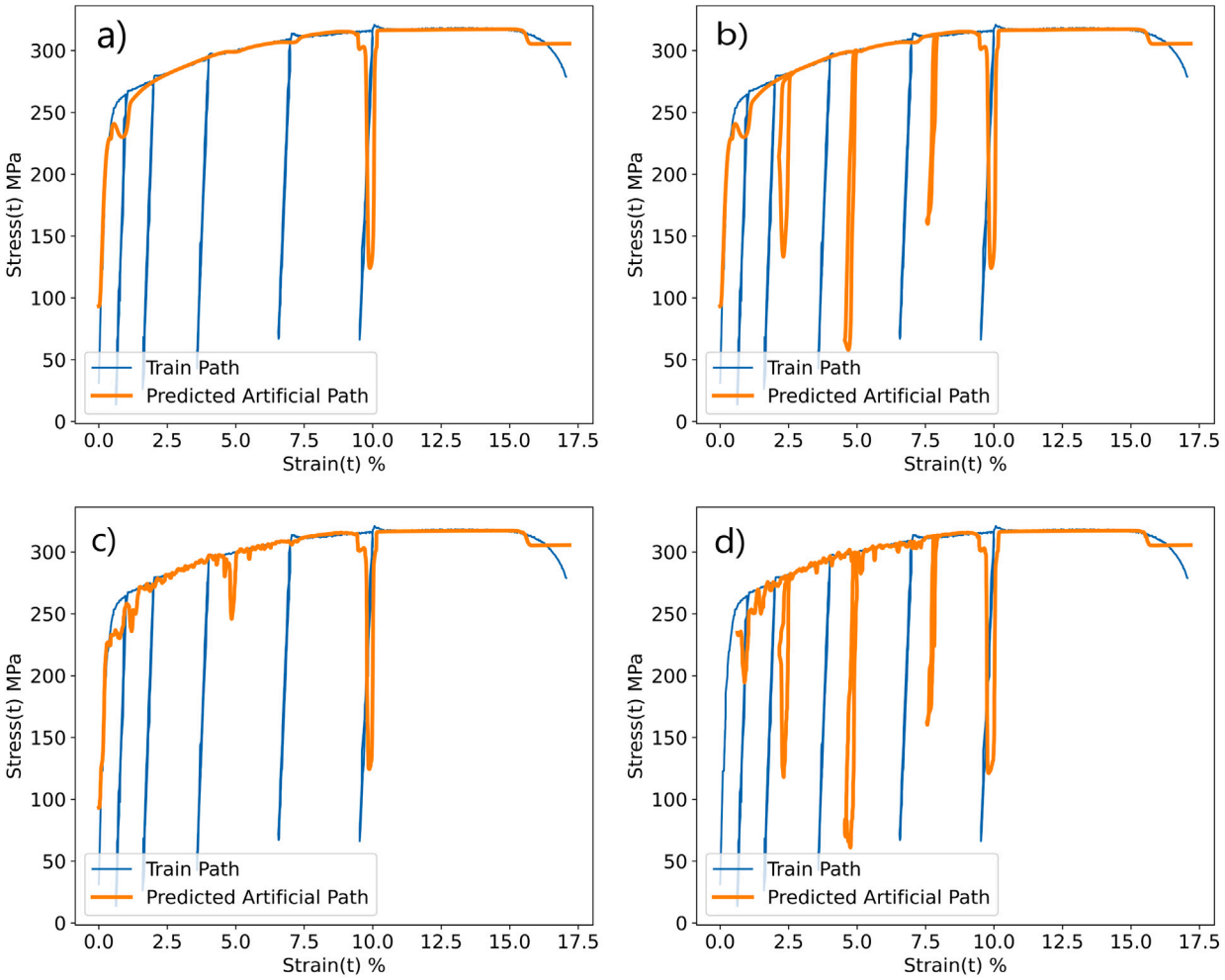


Fig. A.7. (a) LSTM Net prediction on synthetic unseen uniform mono-load path; (b) LSTM Net Prediction on synthetic unseen uniform multi-load path; (c) LSTM Net prediction on synthetic unseen not uniform mono-load path; (d) LSTM Net Prediction on synthetic unseen not uniform multi-load path.

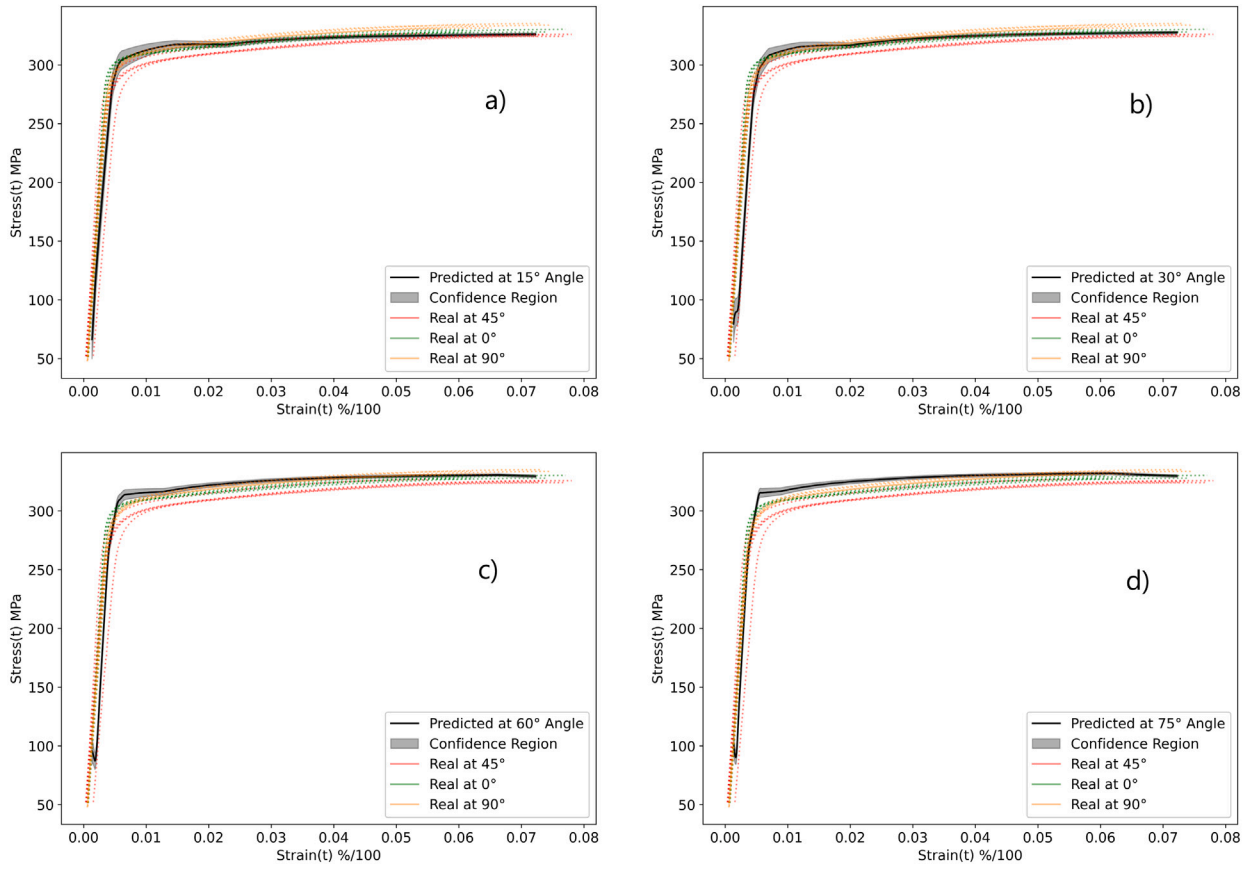


Fig. A.8. Variational Encoder–Decoder prediction on the third dataset evaluation subset in Not Aging conditions. (a) Prediction at 15°; (b) Prediction at 30°; (c) Prediction at 60°; (d) Prediction at 75°.

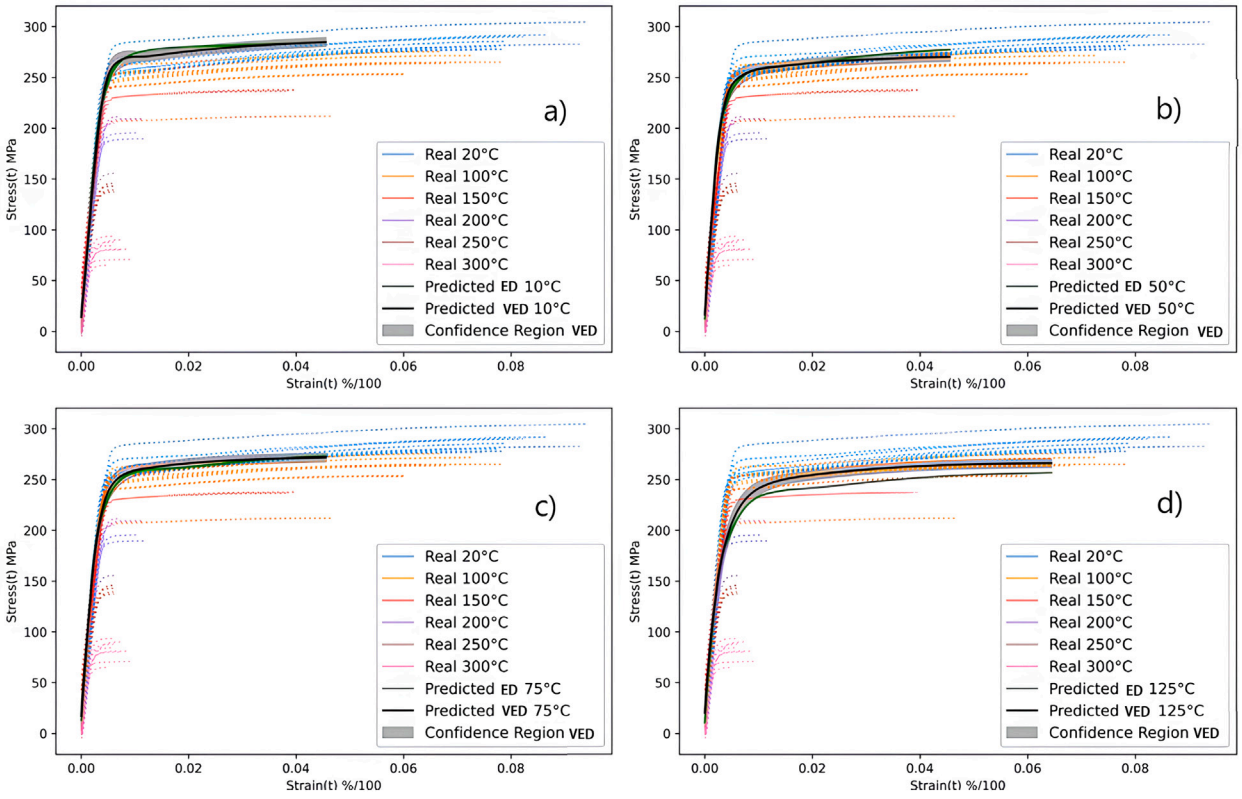


Fig. A.9. Variational (black) and Normal (green) Encoder–Decoder prediction on the fourth dataset evaluation subset. (a) Random composition at 10 °C; (b) Random composition at 50 °C; (c) Random composition at 75 °C; (d) Random composition at 125 °C.

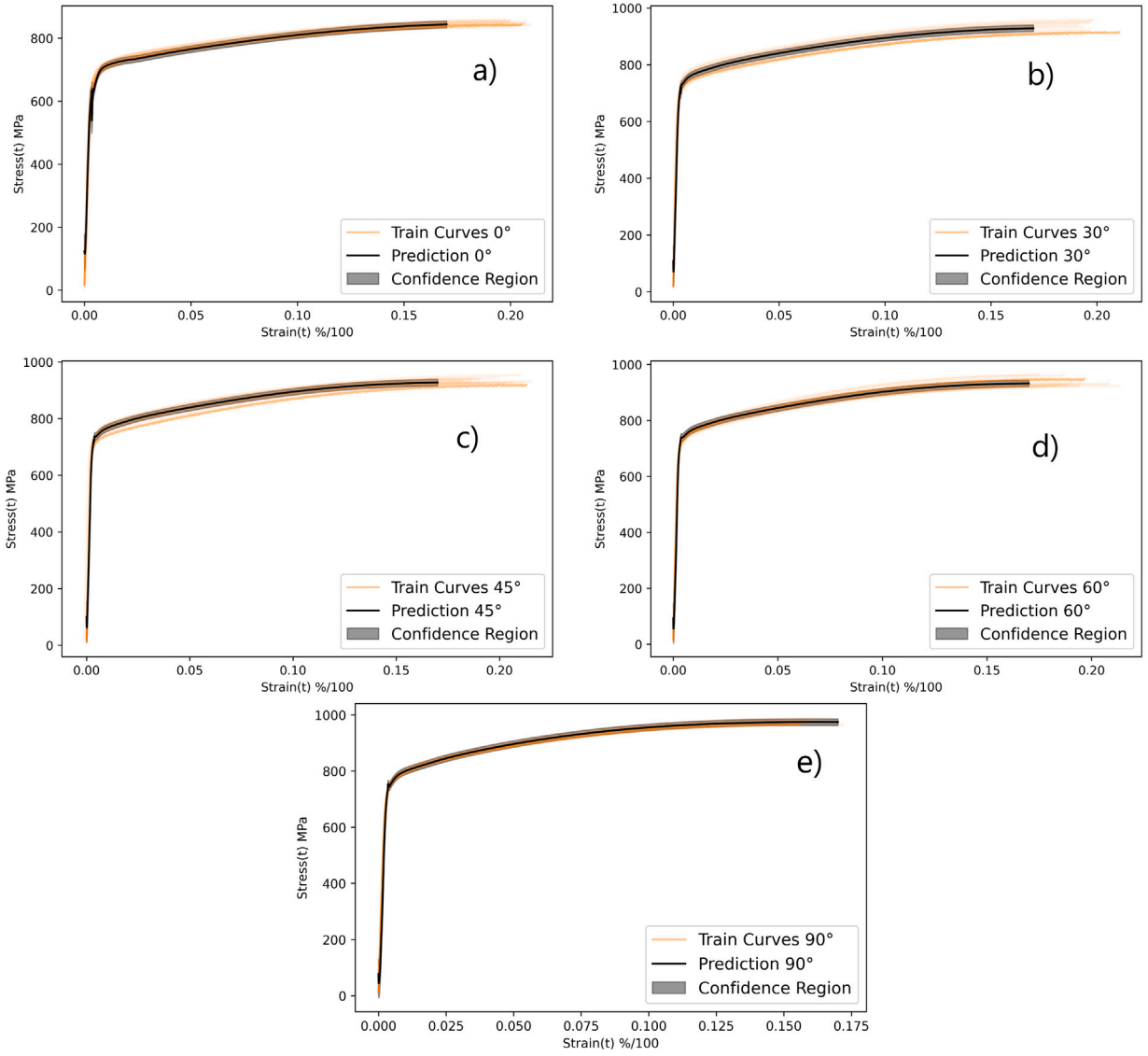


Fig. A.10. Variational Encoder–Decoder prediction on the fifth dataset evaluation subset. Only the angle values existing in the real dataset are shown. Each image has the prediction at a specific angle coupled with ground-truth observations. In order there are predictions at 0° (a), 30° (b), 45° (c), 60° (d), 90° (e). In the subplots it can be seen how the network can correctly model both the shape and the variance of the ground truth observations.

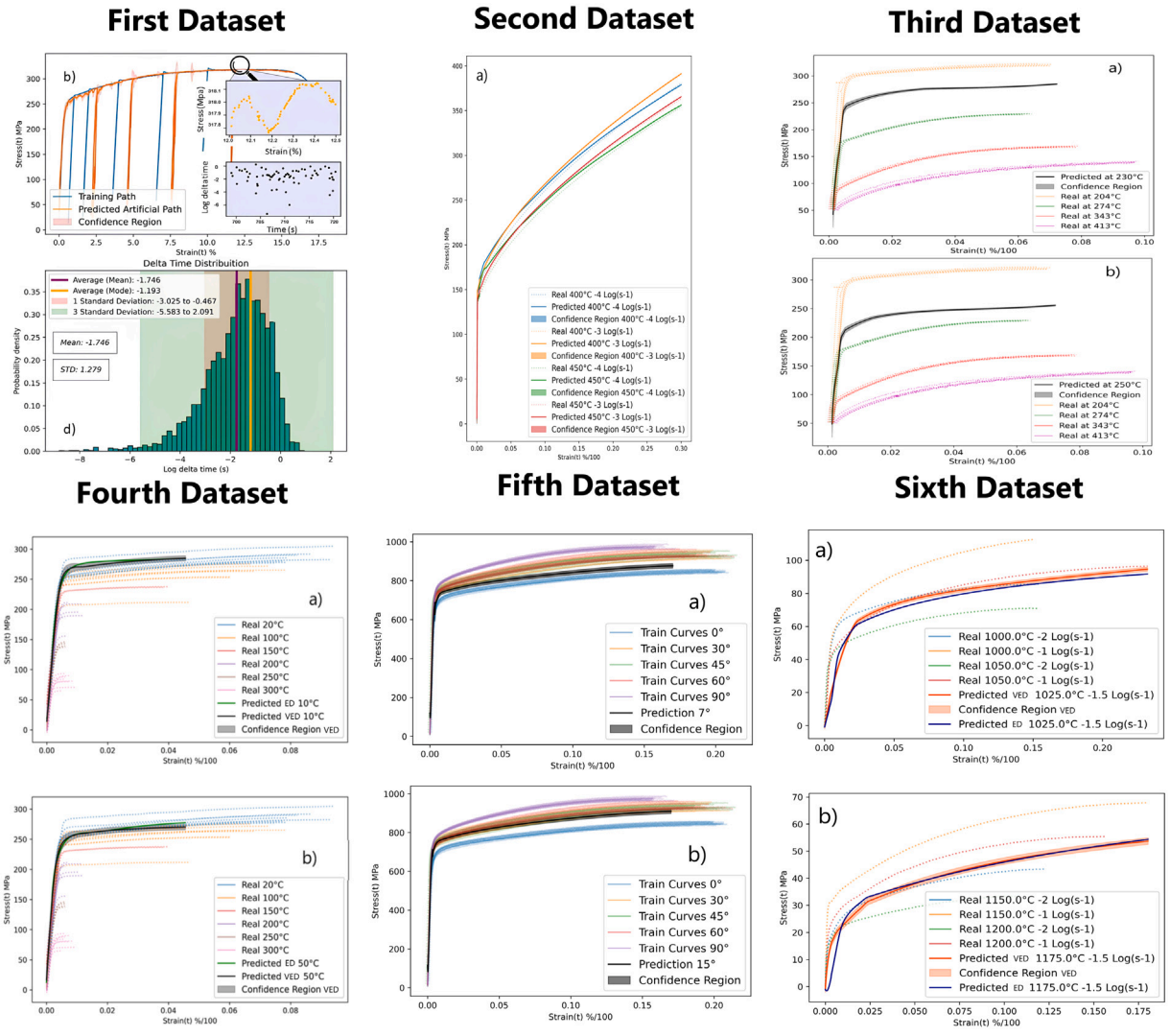


Fig. A.11. Summary of the predictions obtained by both ED and VED on the six datasets. The networks predictions are overall good, despite the heavy and noisy training datasets.

Appendix B. Auxiliary tables

See Tables B.1–B.4.

Table B.1
Third dataset data composition. The Angle of Testing is 0° in the aged temperatures due to the reheating process and subsequent internal stress relaxation that removed any anisotropy from the sample.

Number of curves	Aging temperature	Angles of testing
5	25 °C	45°
5	25 °C	0°
5	25 °C	90°
5	204.4 °C	0° (aged)
5	274 °C	0° (aged)
5	343 °C	0° (aged)
5	413 °C	0° (aged)

Table B.2
Data distribution in the fourth dataset.

Temperature Lot	20 °C	100 °C	150 °C	200 °C	250 °C	300 °C
A	3	3	0	3	0	3
D	3	3	2	3	2	3
G	1	1	1	1	1	1
I	1	1	1	1	1	1

Table B.3
Compositions of the various Lots in the fourth dataset.

Lot	Composition							
	Si	Fe	Cu	Mn	Mg	Cr	Zn	Ti
A	0.56	0.23	0.18	0.07	0.86	0.06	0.02	0.01
D	0.67	0.28	0.23	0.05	0.90	0.05	0.03	0.02
G	0.61	0.15	0.17	0.02	0.88	0.05	0.01	0.01
I	0.68	0.23	0.20	0.04	0.87	0.05	0.02	0.02

Table B.4
Third validation subset composition. Here the composition, strain rate and maximum strain were chosen by sampling from the statistical distribution defined by the training samples.

Temp	10 °C	50 °C	75 °C	125 °C	175 °C
ϵ_{max}	4.55×10^{-2}	4.55×10^{-2}	4.55×10^{-2}	6.45×10^{-2}	2.26×10^{-2}
$\Delta\epsilon$	1.26×10^{-4}	1.26×10^{-4}	1.26×10^{-4}	1.58×10^{-4}	6.88×10^{-5}
Si (%)	0.567	0.664	0.632	0.645	0.562
Fe (%)	0.158	0.263	0.228	0.242	0.153
Cu (%)	0.173	0.222	0.206	0.212	0.171
Mn (%)	0.023	0.063	0.050	0.055	0.021
Mg (%)	0.862	0.895	0.884	0.888	0.861
Cr (%)	0.051	0.059	0.056	0.057	0.050
Zn (%)	0.011	0.027	0.022	0.024	0.010
Ti (%)	0.011	0.019	0.016	0.017	0.010

Temp	225 °C	275 °C	310 °C	330 °C
ϵ_{max}	7.81×10^{-3}	6.05×10^{-3}	6.99×10^{-3}	7.00×10^{-3}
$\Delta\epsilon$	2.04×10^{-5}	2.04×10^{-5}	2.60×10^{-5}	2.60×10^{-5}
Si (%)	0.676	0.660	0.585	0.581
Fe (%)	0.276	0.258	0.178	0.174
Cu (%)	0.228	0.220	0.183	0.181
Mn (%)	0.068	0.061	0.031	0.029
Mg (%)	0.899	0.893	0.868	0.867
Cr (%)	0.060	0.058	0.052	0.052
Zn (%)	0.029	0.026	0.014	0.014
Ti (%)	0.020	0.018	0.012	0.012

Appendix C. Dataset hyperparameters

See Tables C.1 and C.2.

Table C.1
First, Second and Third dataset hyperparameters.

Hyperparameters	First dataset		Second dataset	Third dataset
	VED	ED	VED	VED
motor_neurons	4	8	4	8
total_neurons	35	40	35	25
sparsity	0.303	0.335	0.563	0.422
n_layers_cfc	1	3	1	1
key_size	4	8	4	8
query_size	4	8	4	8
value_size	4	8	4	8
heads	2	2	2	2
dropout	0.0734	0.0728	0.154	0.104
layers_fcn	5	5	4	3
width_fcn	8	128	16	128
decay_fcn	1	1	2	2
norm	True	True	True	True
activation	sigmoid	sigmoid	sigmoid	relu
lr	8.1×10^{-3}	3.7×10^{-3}	7.0×10^{-3}	3.8×10^{-3}
gamma	0.995	0.998	0.981	0.965
epochs	500	500	500	500

Table C.2
Fourth, Fifth and Sixth dataset hyperparameters.

Hyperparameters	Fourth dataset		Fifth dataset	Sixth dataset	
	VED	ED	VED	VED	ED
motor_neurons	4 8	8	6	8	
total_neurons	35	25	25	30	40
sparsity	0.528	0.441	0.413	0.478	0.508
n_layers_cfc	1	1	1	2	2
key_size	4	8	8	6	8
query_size	4	8	8	6	8
value_size	4	8	8	6	8
heads	2	2	2	2	2
dropout	0.124	0.154	0.149	0.043	0.139
layers_fcn	4	6	5	3	3
width_fcn	32	8	64	16	8
decay_fcn	2	1	1	2	1
norm	True	False	True	False	False
activation	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid
lr	5.4×10^{-3}	2.7×10^{-3}	7.8×10^{-3}	7.1×10^{-4}	9.7×10^{-3}
gamma	0.975	0.990	0.966	0.986	0.968
epochs	500	500	500	500	500

References

Aakash, B., Connors, J., Shields, M.D., 2019. Stress-strain data for aluminum 6061-T651 from 9 lots at 6 temperatures under uniaxial and plane strain tension. Data Brief 25, 104085. http://dx.doi.org/10.1016/j.dib.2019.104085.

Alatalo, J., Korpihalkola, J., Sipola, T., Kokkonen, T., 2022. Chromatic and spatial analys. In: Networked Systems. Springer, pp. 303–316. http://dx.doi.org/10.1007/978-3-031-17436-0_20.

ASTM, 2022. Test Methods for Tension Testing of Metallic Materials. ASTM International, http://dx.doi.org/10.1520/e0008_e0008m-22.

Bakhshi, A., Noman, N., Chen, Z., Zamani, M., Chalup, S., 2019. Fast automatic optimisation of CNN architectures for image classification using genetic algorithm. In: 2019 IEEE Congress on Evolutionary Computation. CEC, pp. 1283–1290. http://dx.doi.org/10.1109/CEC.2019.8790197.

Baytas, I.M., Xiao, C., Zhang, X., Wang, F., Jain, A.K., Zhou, J., 2017. Patient subtyping via time-aware LSTM networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 65–74. http://dx.doi.org/10.1145/3097983.3097997.

Benzing, J., Moser, N., Kafka, O., Weaver, J., Derimow, N., Hrabe, N., 2022. AM Bench 2022 challenge Macroscale Tensile Tests at Different Orientations (CHAL-AMB2022-04-MaTTO). National Institute of Standards and Technology, http://dx.doi.org/10.18434/MDS2-2588, URL https://data.nist.gov/od/id/mds2-2588.

Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y., 2018. Recurrent neural networks for multivariate time series with missing values. Sci. Rep. 8 (1), http://dx.doi.org/10.1038/s41598-018-24271-9.

Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D., 2018. Neural ordinary differential equations. http://dx.doi.org/10.48550/ARXIV.1806.07366, arXiv URL https://arxiv.org/abs/1806.07366.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. http://dx.doi.org/10.48550/ARXIV.1406.1078, arXiv URL https://arxiv.org/abs/1406.1078.

Dai, M., Demirel, M.F., Liu, X., Liang, Y., Hu, J.-M., 2023. Graph neural network for predicting the effective properties of polycrystalline materials: A comprehensive analysis. Comput. Mater. Sci. 230, 112461. http://dx.doi.org/10.1016/j.commatsci.2023.112461, URL https://www.sciencedirect.com/science/article/pii/S092702562300455X.

Dao, T., 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning. http://dx.doi.org/10.48550/arXiv.2307.08691, arXiv.

Dao, T., Fu, D.Y., Ermon, S., Rudra, A., Ré, C., 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. http://dx.doi.org/10.48550/arXiv.2205.14135, arXiv.

David, O.E., Greental, I., 2014. Genetic algorithms for evolving deep neural networks. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. Association for Computing Machinery, pp. 1451–1452. http://dx.doi.org/10.1145/2598394.2602287.

DeRose, J.F., Wang, J., Berger, M., 2021. Attention flows: Analyzing and comparing attention mechanisms in language models. IEEE Trans. Vis. Comput. Graphics 27 (2), 1160–1170. http://dx.doi.org/10.1109/TVCG.2020.3028976.

Duan, Y., Zhou, X., Zou, J., Qiu, J., Zhang, J., Pan, Z., 2021. Mask-guided noise restriction adversarial attacks for image classification. Comput. Secur. 100, http://dx.doi.org/10.1016/j.cose.2020.102111, URL https://www.sciencedirect.com/science/article/pii/S0167404820303849.

Galván, E., Mooney, P., 2021. Neuroevolution in deep neural networks: Current trends and future challenges. IEEE Trans. Artif. Intell. 2 (6), 476–493. http://dx.doi.org/10.1109/TAI.2021.3067574.

Gavrilescu, M., Floria, S.-A., Leon, F., Curteanu, S., 2022. A hybrid competitive evolutionary neural network optimization algorithm for a regression problem in chemical engineering. Mathematics 10 (19), http://dx.doi.org/10.3390/math10193581, URL https://www.mdpi.com/2227-7390/10/19/3581.

Gholami, K., Ege, F., Barzegar, R., 2023. Prediction of composite mechanical properties: Integration of deep neural network methods and finite element analysis. J. Compos. Sci. 7 (2), http://dx.doi.org/10.3390/jcs7020054, URL https://www.mdpi.com/2504-477X/7/2/54.

Gorji Maysam, B., Mojitaba, M., Heidenreich Julian, N., Jian, C., Dirk, M., 2020. On the potential of recurrent neural networks for modeling path dependent plasticity. J. Mech. Phys. Solids 143, 103972. http://dx.doi.org/10.1016/j.jmps.2020.103972.

Hasani, R., Lechner, M., Amini, A., Liebenwein, L., Ray, A., Tschaikowski, M., Teschl, G., Rus, D., 2022. Closed-form continuous-time neural networks. Nat. Mach. Intell. 4 (11), 992–1003. http://dx.doi.org/10.1038/s42256-022-00556-7.

Hasani, R., Lechner, M., Amini, A., Rus, D., Grosu, R., 2020. Liquid time-constant networks. http://dx.doi.org/10.48550/ARXIV.2006.04439, arXiv URL https://arxiv.org/abs/2006.04439.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. arXiv:1512.03385.

Hestroffer, J.M., Charpagne, M.-A., Latypov, M.I., Beyerlein, I.J., 2023. Graph neural networks for efficient learning of mechanical properties of polycrystals. Comput. Mater. Sci. 217, 111894. http://dx.doi.org/10.1016/j.commatsci.2022.111894, URL https://www.sciencedirect.com/science/article/pii/S092702562200605X.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hussien, A.G., Abualigah, L., Abu Zitar, R., Hashim, F.A., Amin, M., Saber, A., Almotairi, K.H., Gandomi, A.H., 2019. One pixel attack for fooling deep neural networks. arXiv URL https://www.mdpi.com/2079-9292/11/12/1919.

Iba, H., 2020. Deep Neural Evolution. Springer.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems, Vol. 25. Curran Associates, Inc., URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Lechner, M., Hasani, R., Amini, A., Henzinger, T.A., Rus, D., Grosu, R., 2020. Neural circuit policies enabling auditable autonomy. Nat. Mach. Intell. 2 (10), 642–652. http://dx.doi.org/10.1038/s42256-020-00237-3.

Li, Q.-J., Cinbiz, M.N., Zhang, Y., He, Q., Beausoleil, G., Li, J., 2023a. Robust deep learning framework for constitutive relations modeling. Acta Mater. 254, 118959. http://dx.doi.org/10.1016/j.actamat.2023.118959.

Li, M., Li, S., Tian, Y., Fu, Y., Pei, Y., Zhu, W., Ke, Y., 2023b. A deep learning convolutional neural network and multi-layer perceptron hybrid fusion model for predicting the mechanical properties of carbon fiber. Mater. Des. 227, 111760. http://dx.doi.org/10.1016/j.matdes.2023.111760, URL https://www.sciencedirect.com/science/article/pii/S0264127523001752.

Loshchilov, I., Hutter, F., 2016. CMA-ES for hyperparameter optimization of deep neural networks. http://dx.doi.org/10.48550/arXiv.1604.07269, arXiv.

Marco, M., Chao, G., Filippo, B., 2021. Interlocking mechanism design based on deep-learning methods. Appl. Eng. Sci. 7, http://dx.doi.org/10.1016/j.apples.2021.100056.

Marco, M., Chao, G., Filippo, B., 2022. Predicting stress, strain and deformation fields in materials and structures with graph neural networks. Sci. Rep. 12, http://dx.doi.org/10.1038/s41598-022-26424-3.

Motiwala, A., Soares, S., Atallah, B.V., Paton, J.J., Machens, C.K., 2022. Efficient coding of cognitive variables underlies dopamine response and choice behavior. Nature Neurosci. 25 (6), 738–748. http://dx.doi.org/10.1038/s41593-022-01085-7.

Ning, M., Huaixian, Y., Kai, W., 2023. Prediction of the remaining useful life of supercapacitors at different temperatures based on improved long short-term memory. Energies 16 (14), http://dx.doi.org/10.3390/en16145240.

Niu, Z., Zhong, G., Yu, H., 2021. A review on the attention mechanism of deep learning. Neurocomputing 452, 48–62. http://dx.doi.org/10.1016/j.neucom.2021.03.091, URL https://www.sciencedirect.com/science/article/pii/S092523122100477X.

Press, W.H., Teukolsky, S.A., 1990. Savitzky-golay smoothing filters. Comput. Phys. 4 (6), 669. http://dx.doi.org/10.1063/1.4822961.

Python, 2023a. Python 3.10. URL https://www.python.org/.

Pytorch, 2023b. Pytorch 1.13.1. URL https://pytorch.org/.

Rumelhart, D.E., McClelland, J.L., 1987. Learning internal representations by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations. MIT Press, pp. 318–362, URL http://ieeexplore.ieee.org/document/6302929.

Sen, J., Dasgupta, S., 2023. Adversarial attacks on image classification models – FGSM and patch attacks and their impact. http://dx.doi.org/10.48550/arXiv.2307.02055.

Su, J., Vargas, D.V., Sakurai, K., 2019. One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. 23 (5), 828–841. http://dx.doi.org/10.1109/TEVC.2019.2890858, URL https://ieeexplore.ieee.org/document/8601309.

Sun, Y., Xue, B., Zhang, M., Yen, G.G., 2019. Automatically evolving CNN architectures based on blocks. http://dx.doi.org/10.48550/arXiv.1810.11875, arXiv.

Sun, Y., Xue, B., Zhang, M., Yen, G.G., Lv, J., 2020. Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE Trans. Cybern. 50 (9), 3840–3854. http://dx.doi.org/10.1109/tcyb.2020.2983860.

Tomczak, J.M., 2022. Deep Generative Modeling. Springer.

Umbrello, D., M'Saoubi, R., Outeiro, J., 2007. The influence of Johnson–Cook material constants on finite element simulation of machining of AISI 316l steel. Int. J. Mach. Tools Manuf. 47 (3–4), 462–470. http://dx.doi.org/10.1016/j.ijmachtools.2006.06.006.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. http://dx.doi.org/10.48550/ARXIV.1706.03762, arXiv URL https://arxiv.org/abs/1706.03762.

Vode, F., Malej, S., Arh, B., Tehovnik, F., Podgornik, B., 2019. Description of hot compressive stress-strain curves using transfer functions. Metals 9 (3), 290. http://dx.doi.org/10.3390/met9030290.

Wandb, 2023c. Weight and biases (wandb). URL https://wandb.ai/site.

Weaver, J.S., Khosravani, A., Castillo, A., Kalidindi, S.R., 2016. High throughput exploration of process-property linkages in Al-6061 using instrumented spherical microindentation and microstructurally graded samples. Integr. Mater. Manuf. Innov. 5 (1), 192–211. http://dx.doi.org/10.1186/s40192-016-0054-3.

Wu, H., Xu, J., Wang, J., Long, M., 2022. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. http://dx.doi.org/10.48550/arXiv.2106.13008, arXiv.

Zheng, X., Zheng, P., Zhang, R.-Z., 2018. Machine learning material properties from the periodic table using convolutional neural networks. Chem. Sci. 9, 8426–8432. http://dx.doi.org/10.1039/C8SC02648C.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W., 2020. Informer: Beyond efficient transformer for long sequence time-series forecasting. http://dx.doi.org/10.48550/arXiv.2012.07436, arXiv.