# Real-Time Simulation of Plastic Deformation Using Long Short-Term Memory Multi-Task Neural Networks

Schmeitz, Ruben

*Award date:*
2023

# Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

18-10-2022

Name

R. Schmeitz

ID-number

1233052

Signature

*Submit the signed declaration to the student administration of your department.*

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

# Real-Time Simulation of Plastic Deformation Using Long Short-Term Memory Multi-Task Neural Networks

Master Thesis
2023


| Name | Student number |
|---|---|
| R.Schmeitz | 1233052 |

Supervisors:
prof.dr.ir. O. (Olaf) van der Sluis (TU/e & Philips Research)
dr.ir. J.C. (Joris) Remmers (TU/e & EAISI)


Eindhoven, Oktober 2, 2023

# Preface

I would like to extend my gratitude to my thesis supervisors, Olaf van der Sluis and Joris Remmers, for their invaluable guidance and support throughout this project. Their expertise in computational mechanics and dedication to innovation inspired me to explore novel approaches for real-time simulation.

This research was conducted in collaboration with Philips, who provided the motivation for developing an efficient technique to model catheter deformations. Catheters possess complex geometries and exhibit nonlinear path-dependent behavior that pose computational challenges. My prior internship experience at Philips, working on predicting beam deformations under the tutelage of Bart and Marco, fueled my drive to investigate this problem using a different methodology. I'm grateful to Marco Baragona and Bart Bakker from Philips for sharing their guidance during my internship.

Building on my previous work, this project aims to propose a multi-task recurrent neural network framework for surrogate modeling of elastoplasticity. The approach combines dimensionality reduction, long short-term memory networks, and multi-task learning to achieve speedups while retaining accuracy. I appreciate Olaf for presenting me with this stimulating problem and granting me the opportunity to experiment with innovative solutions.

It's been a privilege to conduct this research, and I look forward to starting my Ph.D. under Joris's supervision. I hope that my work contributes to tackling complex simulations in modern biomedical technologies. With the knowledge gained, I aim to drive future innovations.

This master thesis report is written in a paper format given that we intend to publish the work in the future.

# Real-Time Simulation of Plastic Deformation Using Long Short-Term Memory Multi-Task Neural Networks

Ruben Schmeitz[1*], Olaf van der Sluis[1,2†] and Joris J.C. Remmers[1,2,3†]

[1*]Department Mechanical Engineering, Eindhoven University of Technology,Eindhoven, The Netherlands.
[2]Philips Research, High Tech Campus 4, 5654 AE, Eindhoven, The Netherlands.
[3]Eindhoven AI Systems Institute, Laplace 32, 5612 AJ, Eindhoven, The Netherlands.

*Corresponding author(s). E-mail(s): r.schmeitz@tue.nl ;
Contributing authors: o.v.d.sluis@tue.nl; j.j.c.remmers@tue.nl;
†These authors contributed equally to this work.

**Abstract**

Accurately modeling deformations in real-time is essential for applications like minimally invasive therapies and robotics, but computational challenges persist. This study introduces a novel surrogate modeling framework that combines proper orthogonal decomposition (POD), long short-term memory (LSTM) networks, and multi-task learning to achieve efficient yet accurate real-time predictions. A notable feature is the integration of multi-task learning, which contributes to a more holistic comprehension of the system's overall state. The approach leads to improved predictive accuracy while also accelerating computations. The model has been validated on 2D elastoplastic examples to demonstrate the model's performance, achieving less than 1% error in predicting path-dependent variables like plastic strain in less than 1 millisecond.

**Keywords:** Surrogate modeling, Path-dependent deformation, Long Short-Term Memory, Proper Orthogonal Decomposition

## 1 Introduction

Flexible instruments are commonly utilized in various medical procedures such as diagnostic and interventional purposes within sophisticated anatomical structures [1]. Nevertheless, accurately modeling the complex path-dependent deformations of these instruments remains a computational challenge due to the nonlinearity of materials and complex geometries involved. The finite element method (FEM) is a widely recognized approach in solid mechanics, renowned for its capacity to address boundary value problems systematically [2]. Nevertheless, as spatial discretization is refined to enhance accuracy, the computational demands of

FEM increase significantly. This escalation, compounded by the complexities introduced by material nonlinearity, limits its practicality for real-time applications, such as the prediction of the deformation of flexible instruments. The ability to make real-time predictions holds immense potential for critical applications where a Digital Twin is needed [3], including surgical assistance, preoperative planning, and virtual training systems [4]. By offering swift yet precise approximations of instrument deformation, this technology can empower medical professionals to anticipate device-tissue interactions, thereby facilitating precise navigation and minimizing the risk of damage to sensitive tissue or the instrument itself.

In response to these limitations, Projection-based reduction techniques, such as Proper Orthogonal Decomposition (POD), have emerged as valuable solutions [5]. These methods effectively reduce the dimensionality of full order data, leading to lower computational costs while maintaining high accuracy [6]. For linear boundary value problems, which do not require iterative computations, projection-based reduction methods offer an efficient means of reducing costs without compromising accuracy significantly. Unfortunately, this approach is less efficient for nonlinear systems, that require iterative computations, such as the Newton-Raphson method, due to the necessity of constructing system matrices and performing inverse calculations. When history-dependent material behavior is involved, solving boundary value problems becomes computationally intractable due to the need to perform calculations for all loading paths. Even when reduction techniques are employed, nonlinear analysis remains challenging and intricate [7, 8]. To address these challenges, hyper-reduction methods have been developed. These methods aim to alleviate the computational costs associated with nonlinear terms in large systems by strategically selecting a small set of nodes (or elements) in the mesh over which the nonlinearity is evaluated [9]. This approach efficiently interpolates the nonlinearity over the remaining mesh, reducing computational demands.

Despite progress in reduced-order modeling methods, nonlinear systems still present substantial computational challenges. This is where the potential breakthrough offered by machine learning models comes into play, promising to provide an effective means of capturing the complex behavior of such highly nonlinear systems. By optimizing the parameters of a neural network (NN) during a training procedure, these models can accurately establish the mapping between input and desired output [10]. Incorporating activation functions on the NN's neurons enables these mappings to effectively approximate highly nonlinear functions. This promising approach may pave the way for tackling the challenges associated with nonlinear analysis, particularly in the presence of history-dependent material behavior. By leveraging machine learning, it becomes possible to overcome the computational bottlenecks that hinder traditional techniques and potentially open new avenues for more efficient and accurate simulations.

Notably, Mendizabal [11] introduced the U-mesh model, which utilizes both full-order data and a U-net neural network to predict nonlinear deformations in hyperelastic materials. This model is trained on finite element simulation data to acquire knowledge of the deformation, enabling rapid and accurate predictions of the complete 3D displacement field. Additionally, Odot et al. [12] proposed the DeepPhysics model, an extension of the U-net framework that incorporates a Bayesian approach to address the complexities associated with capturing nonlinear material behavior. El Said [13] presented a deep learning framework employing Deep Recurrent Convolutional Neural Networks to forecast the nonlinear response of composite materials. This technique has demonstrated effectiveness in predicting complete 3D stress-strain curves from images of Representative Volume Elements (RVE), facilitating efficient nonlinear analysis.

These machine learning models have shown promise in capturing linear and nonlinear behavior using full-order data. However, the computational expense of training these models intensifies as the mesh size expands, due to an increase in the number of trainable parameters. This limits the scalability that restricts the application to larger systems.

To overcome this challenge, projection-based reduction can be incorporated. This reduced-order data can be obtained through the proper orthogonal decomposition (POD) method. By harnessing the power of POD,

the elimination of the least significant modes becomes possible, resulting in a significant reduction of the required number of degrees of freedom. Brunton [14] demonstrated the efficiency of combining POD with machine learning in the field of fluid dynamics. In the field of solid mechanics, Im et al. [15] recently proposed using POD-assisted long short-term memory (LSTM) networks to model elastoplastic deformations. Their approach trains separate LSTM models to predict different field variables from POD compressed data for a specific problem geometry. While showing good accuracy, their methodology lacks a shared representation across tasks and requires training individual models for each problem setup.

This provides motivation for exploring multi-task neural network architectures that leverage shared representations to enhance generalization across field variables and problem configurations. Shared layers may enable more comprehensive spatio-temporal modeling while task-specific outputs retain flexibility. This study explores this multi-task approach for surrogate modeling of nonlinear elastoplasticity using reduced-order data.

This innovation in architecture allows for the capture of a comprehensive spatio-temporal representation of the sequence data using shared LSTM layers, directing this representation towards separate dense layers tailored for specific field variables, such as displacement or plastic strain. By utilizing shared layers, the model learns universal features, while the task-specific layers maintain the flexibility to adapt to each unique field variable. This multi-task structure effectively reduces the overall number of parameters compared to using separate LSTMs for each field. Furthermore, the connection between shared layers not only enhances generalization but also acts as a form of regularization, which contributes to improved performance and stability.

The primary objective of this paper is to demonstrate the capability of the multi-task neural network framework to learn universal representations, enabling accurate prediction across diverse field variables. This adaptability and precision form the foundation for tackling complex challenges across various domains, especially in medical device simulations.

To achieve this goal, the paper first explores fundamental theoretical concepts, including the finite element method, elastoplasticity, data reduction, and neural networks. Building on this basis, the LSTM multi-task framework is presented, elucidating its architecture, training methodology, and evaluation metrics. The framework's validation entails a thorough analysis of two distinct 2D use cases characterized by small strain and elastoplasticity: a table model and a cantilever beam model. For each case, we delve into the setup, data collection, data reduction techniques, and presentation of quantitative results. With the efficacy of the framework established, discussions shift to an examination of the performance of the multi-task architecture. Limitations are identified and recommendations for future research directions are proposed. Overall, this structured narrative showcases the promising performance of the multi-task framework, laying the groundwork for tackling complex 3D problems in real-time medical device simulations.

## 2 Physics Based Model

In order to develop an efficient and accurate surrogate model using machine learning, it is first necessary to generate training data through finite element simulations. However, the raw simulation data often has very high dimensionality corresponding to the degrees of freedom in the finite element mesh. To obtain a more compact and optimal dataset to feed into the machine learning model, techniques like POD can be applied to reduce the dimensionality of the original data while retaining most of the essential information. The following section provides an overview of key aspects of the finite element method relevant to this study and discusses the use of POD for dimensionality reduction of the simulation data. This reduced-order data will form the input for training the proposed neural network surrogate model.

## 2.1 Finite Element Method

The FEM is a numerical technique widely used to solve boundary value problems (BVPs). It involves several key equations and steps that enable the accurate approximation of solutions to a wide range of problems, considering complex geometries, arbitrary boundary conditions, and varying material properties. However, solving a BVP using FEM can be challenging due to various factors, with material behavior being a crucial one. The complexity of the problem can significantly vary depending on the material model employed. For instance, a purely elastic material model, which lacks dependencies on loading history, is simpler to solve compared to an elastoplastic material model with kinematic hardening. In the context of this paper, we specifically utilize an elastoplastic material with kinematic hardening.

We consider a domain $\Omega$ subjected to displacements at boundary $\Gamma_d$ and tractions at boundary $\Gamma_t$, as depicted in Figure 1.

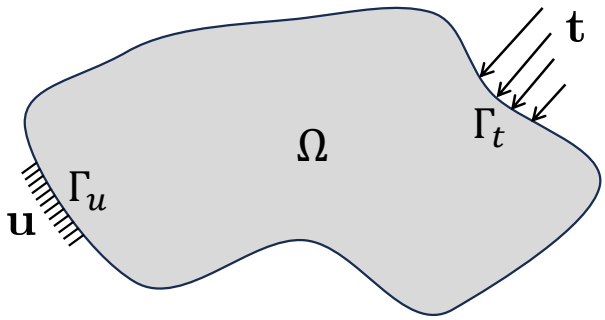



**Fig. 1**: Boundary Value Problem

Neglecting the influence of body forces, the strong form of the equilibrium can be formulated as:

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} = \mathbf{0} \tag{1}$$

Here, $\boldsymbol{\sigma}$ represents the Cauchy stress tensor, a symmetric tensor. We make an assumption known as the kinematic relation for small strains, often referred to as infinitesimal strain theory. This assumption implies that the strains in the material are sufficiently small to consider the undeformed and deformed configurations as geometrically similar:

$$\boldsymbol{\varepsilon} = \boldsymbol{\nabla}^s \mathbf{u} \tag{2}$$

Here, $\boldsymbol{\nabla}^s$ is the symmetric gradient operator. By applying Gauss's theorem and discretizing, the linear momentum balance can be expressed as:

$$\int_\Omega \boldsymbol{B}^T \boldsymbol{\sigma} \, \mathrm{d}\Omega = \int_{\Gamma_t} \boldsymbol{N}^T \boldsymbol{t} \, \mathrm{d}\Gamma \tag{3}$$

In this equation, $\boldsymbol{N}$ represents the element shape functions, describing the displacement field the domain in terms of nodal displacements. Typically, these shape functions take the form of low-order polynomials. The relationship between strains $\boldsymbol{\varepsilon}$ and nodal displacements $\boldsymbol{u}$ is facilitated through the strain-displacement matrix $\boldsymbol{B}$:

$$\boldsymbol{\varepsilon} = \boldsymbol{B}\boldsymbol{u} \tag{4}$$

It is important to highlight that the constitutive stress-strain relationship is nonlinear for elastoplastic materials. This nonlinearity stems from the complex plastic flow behavior that depends on the current state and loading history. Owing to this nonlinearity, an iterative numerical solution approach becomes imperative for solving Equation (3):

$$\boldsymbol{K}\Delta\boldsymbol{u} = \boldsymbol{f}^{\text{ext}} - \boldsymbol{f}^{\text{int}} \tag{5}$$

In this equation, $\boldsymbol{K}$ characterizes the connection between changes in nodal displacements $\Delta\boldsymbol{u}$ and can be determined as:

$$\boldsymbol{K} = \int_{\Omega} \boldsymbol{B}^T \boldsymbol{D}\boldsymbol{B}\mathrm{d}\Omega \tag{6}$$

Here, $\boldsymbol{D}$ represents the consistent tangent operator, defined as:

$$\boldsymbol{D} = \frac{\partial\boldsymbol{\sigma}}{\partial\boldsymbol{\varepsilon}} \tag{7}$$

The external force $\boldsymbol{f}^{\text{ext}}$ and the internal force $\boldsymbol{f}^{\text{int}}$ can be expressed as follows:

$$\boldsymbol{f}^{\text{int}} = \int_{\Omega} \boldsymbol{B}^T\boldsymbol{\sigma}\,\mathrm{d}\Omega \tag{8}$$

$$\boldsymbol{f}^{\text{ext}} = \int_{\Gamma_t} \boldsymbol{N}^T\boldsymbol{t}\,\mathrm{d}\Gamma \tag{9}$$

For more in-depth information, please consult the work by Borst et al. [16].

## 2.2 Elastoplasticity

The constitutive behavior of elastoplastic materials considers both elastic strain ($\boldsymbol{\varepsilon}^e$) and plastic strain ($\boldsymbol{\varepsilon}^p$), which sum up to the total strain:

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p \tag{10}$$

In contrast to elastic strain, plastic strain signifies an irreversible process resulting from the movement of dislocations facilitated by the stress at the yield point [17]. In the case of plastic flow, the plastic strain rate can be expressed as:

$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\frac{\partial f_y}{\partial\boldsymbol{\sigma}} = \dot{\lambda}\boldsymbol{n} \tag{11}$$

Here, $\dot{\lambda}$ determines the magnitude of the plastic flow, $\boldsymbol{n}$ the relative magnitudes of the components of the plastic flow, and $f_y$ the yield function.
After a plastic strain increment, the stress state needs to be returned back onto the yield surface. This is achieved using a return mapping algorithm that solves for the plastic multiplier $\lambda$ that will return the stress to the yield surface after plastic flow [16]. The return mapping algorithm enforces the stresses staying on or within the yield surface after plastic flow, providing a critical coupling between the elastic stress increments and plastic flow.

## 2.3 Hardening

Plastic deformation manifests subsequent to surpassing the yield point. The yield point is defined by the yield function expressed as:

$$f_y(\boldsymbol{\sigma}, \kappa) \leq 0 \tag{12}$$

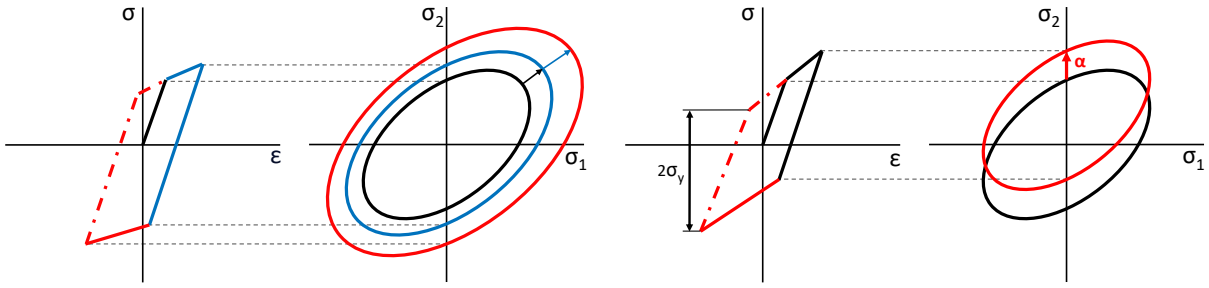
Here, $\kappa$ signifies the plastic hardening parameter, a scalar value contingent upon the strain history. The work-hardening assumption for $\kappa$ can be articulated as follows:

$$\dot{\kappa} = \frac{\boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}^p}{\sigma_y} \tag{13}$$

Here, $\sigma_y$ represents the yield stress. The hardening parameter can be determined by integrating $\dot{\kappa}$ along the loading path:

$$\kappa = \int \dot{\kappa} \mathrm{d}t \tag{14}$$

Since the yield function solely depends on its loading history through a scalar-valued parameter, the yield surface will expand or contract solely in the stress space, as illustrated in Figure 2a. This phenomenon is commonly known as isotropic hardening.




(a) Isotropic hardening

(b) Kinematic hardening

**Fig. 2**: Types of hardening in 2D

On the other hand, Figure 2b depicts kinematic hardening, which involves multiple translations in the stress space. This type of hardening effectively captures the effects observed during cyclic loading, commonly referred to as the Bauschinger effect in metals. The translation of the yield surface can be described by the back stress tensor $\boldsymbol{\alpha}$. Its hardening law is defined as:

$$\dot{\boldsymbol{\alpha}} = \frac{C}{\sigma_y}(\boldsymbol{\sigma} - \boldsymbol{\alpha})\dot{\bar{\varepsilon}}^p - \gamma \boldsymbol{\alpha} \dot{\bar{\varepsilon}}^p \tag{15}$$

Here, $H$ represents the kinematic hardening modulus, $\dot{\bar{\varepsilon}}^p$ denotes the equivalent plastic strain rate, and $\gamma$ a scalar parameter that adjusts the influence of isotropic hardening on the material. Assuming a von Mises yield function incorporating hardening behavior leads to:

$$f_y = \frac{1}{2}(\boldsymbol{\sigma}' - \boldsymbol{\alpha})^T(\boldsymbol{\sigma}' - \boldsymbol{\alpha}) - \frac{1}{3}\sigma_y^2 = 0 \tag{16}$$

In this equation, $\boldsymbol{\sigma}'$ represents the deviatoric stress tensor. Using the elastic tangential stiffness tensor $\boldsymbol{D}^e$ the consistent tangent can be formulated as:

$$\boldsymbol{D} = \begin{cases} \boldsymbol{D}^e & \text{if } f_y < 0 \\ \boldsymbol{D}^e - \dfrac{\boldsymbol{D}^e \dfrac{\partial f_y}{\partial \boldsymbol{\sigma}} \left(\dfrac{\partial f_y}{\partial \boldsymbol{\sigma}}\right)^T \boldsymbol{D}^e}{\left(\dfrac{\partial f_y}{\partial \boldsymbol{\sigma}}\right)^T \boldsymbol{D}^e \left(\dfrac{\partial f_y}{\partial \boldsymbol{\sigma}}\right)} - \dfrac{\Delta\kappa}{\Delta\lambda}\dfrac{\partial f_y}{\partial \kappa} & \text{if } f_y = 0 \end{cases} \tag{17}$$

## 2.4 Data Reduction

In previous studies on predicting the deformation of various boundary value problems, the conventional approach has involved utilizing full-order data [11, 12]. This dataset comprises the specified field variable at each node, accounting for all principal directions. Typically, Equation (5) is solved, followed by the integration of variables such as stress and strain at the integration points of the elements. However, displacements are assessed at the nodes. To ensure consistency, the stress and strain values are extrapolated at these nodes in our approach. By collecting this nodal data across each time step, we construct a high-dimensional dataset denoted as $\boldsymbol{S} \in \mathbb{R}^{n \times m}$ for a specific field variable. Here, $n$ represent the total number of degrees of freedom (DOF) in the case of displacement. In the case of the specific field variable, this corresponds to the total number of dimensions, which is equivalent to the number of nodes multiplied by the dimensionality. Meanwhile, $m$ indicates the total number of loading steps.

In this context, Principal Component Analysis (PCA) or Proper Orthogonal Decomposition (POD) techniques can be applied to reduce the dimensionality of the dataset and make the neural network independent of the mesh, ultimately reducing the number of trainable parameters involved.

To reduce the dimensionality of $\boldsymbol{S}$, which describes the evolution of a specific field variable over time, we first decompose $\boldsymbol{S}$ into a mean vector $\bar{\boldsymbol{S}} \in \mathbb{R}^n$ which takes the mean over the columns, and its standard deviation $\boldsymbol{S}^{\text{dev}} \in \mathbb{R}^{n \times m}$ as follows:

$$\boldsymbol{S}^{\text{dev}} = \boldsymbol{S} - \bar{\boldsymbol{S}} \tag{18}$$

By subtracting the mean vector, the data for the field variable are centered, meaning that the principal components do not include the offset at each time step from the center. The next step involves factorizing $\boldsymbol{S}^{\text{dev}}$ using singular value decomposition (SVD) as follows:

$$\boldsymbol{S}^{\text{dev}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} \tag{19}$$

Here, $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ represents a matrix with left singular vectors, $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times m}$ is a diagonal matrix with singular values arranged in descending order, and $\boldsymbol{V} \in \mathbb{R}^{m \times m}$ represents the right singular vectors. Both $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices. While $\boldsymbol{U}$ contains the singular vectors that represent spatial modes, $\boldsymbol{V}$ represents the temporal modes.

Since $\boldsymbol{U}$ contains singular values in descending order, we can neglect the least significant ones based on a specific criteria. This results in a truncated version of $\boldsymbol{U}$ called the transformation matrix $\boldsymbol{T} \in \mathbb{R}^{r \times n}$. Subsequently, the transformation matrix can be used to compute the corresponding Eigenvalue matrix according to:

$$\boldsymbol{A} = \boldsymbol{T} \cdot \boldsymbol{S}^{\text{dev}} \tag{20}$$

The resulting matrix $\boldsymbol{A} \in \mathbb{R}^{r \times m}$ has $r$ dominant vectors in its rows representing the number of Eigenvalues and columns representing the evolution of these values over time. To obtain the recovered matrix $\boldsymbol{S}^{\text{rec}}$, the inverse process can be applied:

$$\boldsymbol{S}^{\text{rec}} = \boldsymbol{T}\boldsymbol{A} + \bar{\boldsymbol{S}} \tag{21}$$

The determination of the number of dominant Eigenvalues, involves incrementally increasing its value until the normalized Root Mean Square Error (RMSE) between the original data matrix $\boldsymbol{S}$ and the reconstructed data matrix $\boldsymbol{S}^{\text{rec}}$ falls below a specified threshold:

$$e_{\text{RMSE\%}} = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{\boldsymbol{S}_i^{\max}}\sqrt{\frac{\sum_{i=1}^{N}(\boldsymbol{S}_{ij}^{\text{rec}} - \boldsymbol{S}_{ij})^2}{N}}\right) \times 100\% \tag{22}$$

Here, $\boldsymbol{S}_i^{\max}$ represents the maximum value in each row. The subscripts $(i, j)$ denote the elements of their respective matrices.

The final $r$ value that meets this criterion represents the number of dominant Eigenvalues, yielding an accurate low-dimensional approximation of the original data. Retaining too few Eigenvalues would lose critical information, while retaining too many would include noise. The selection of these dominant vectors is not arbitrary, further underscoring the method's non-greedy nature.

# 3 Machine Learning Framework

Through the incorporation of the modes $\boldsymbol{T}$, the mean vector denoted as $\bar{\boldsymbol{S}}$, and their corresponding Eigenvalues $\boldsymbol{A}$, it becomes possible to portray each discrete timestep within the complete dataset, $\boldsymbol{S}$, while preserving a substantial portion of the underlying information. This paves the way for the integration of a machine learning model as a surrogate, effectively interpolating $\boldsymbol{A}$ by establishing a mapping from the input, denoted as a set of boundary conditions $\boldsymbol{X}_t$ for a single loading step $t$, to the corresponding Eigenvalues $\boldsymbol{A}_t$. The mapping is made by training the neural networks hyperparameters $\theta$.

$$f_{\mathrm{NN}}(\boldsymbol{X}_t, \boldsymbol{H}_{t-1}, \boldsymbol{\theta}) = \boldsymbol{A}_t \tag{23}$$

In this context, $\boldsymbol{A}_t$ corresponds to the column at time step $t$ in the matrix $\boldsymbol{A}$. The vector $\boldsymbol{H}_{t-1}$ represents the historical state vector from the preceding time step, encompassing both the cell state $\boldsymbol{c}_{t-1}$ and the hidden state $\boldsymbol{h}_{t-1}$, as detailed in the following sections. When the changing boundary condition entails a point force with varying magnitude and position, $\boldsymbol{X}$ assumes the following configuration:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{P} \end{bmatrix} \tag{24}$$

Here, $\boldsymbol{F} \in \mathbb{R}^{d \times m}$ represents the magnitude of the force, and $\boldsymbol{P} \in \mathbb{R}^{d \times m}$ denotes the position of the force, $d$ denotes the number of dimensions pertinent to the problem, which is 2 in the case of a 2D problem.

## 3.1 Neural Network

The mapping is established using a Deep Neural Networks (DNN). DNNs are composed of an input layer, hidden layers, and output layers, each consisting of multiple nodes. These nodes are interconnected across layers through weights. The output signal of the current nodes in a layer is determined by the weights and biases, the nodes from the previous layer, and the activation function. The activation function introduces non-linearity to the output. Mathematically, a layer can be represented as:

$$f_i(\boldsymbol{x}) = a(\boldsymbol{W}_i \boldsymbol{x} + \boldsymbol{b}_i) \tag{25}$$

Here, $\boldsymbol{x}$ represents the output from the previous layer, $\boldsymbol{W}_i$ corresponds to the connection weights, $\boldsymbol{b}_i$ denotes the biases, and $a$ signifies the activation function. The inclusion of multiple layers leads to the following expression:

$$f(\boldsymbol{x}) = g \circ f_K \circ .... \circ f_2 \circ f_1(\boldsymbol{x}) \tag{26}$$

Here, $g$ represents the output layer, $k$ the number of layers, and $\circ$ the mathematical operation of function composition. Once the weights and biases have been initialized following a specific condition, such as Xavier or orthogonal initialization [18], the subsequent stages involve training the neural network through the following steps:

1. **Forward Propagation:**
   In this step, the input data is fed into the network, propagating it forward through the layers.

2. **Loss Calculation:**
   A loss function is employed to measure the discrepancy between the predicted output and the desired output for a given input. In this case, the mean squared error (MSE) loss function is utilized, as specified in Equation (27).

$$\mathrm{MSE}(\boldsymbol{\theta}) = \frac{1}{m \times r} \sum_{t=1}^{\mathrm{m}} \sum_{i=1}^{r} \left( \boldsymbol{A}_{it} - \hat{\boldsymbol{A}}_{it}(\boldsymbol{\theta}) \right)^2 \tag{27}$$

In this equation, $\boldsymbol{\theta}$ encompasses the neural network's parameters, including weights and biases. $\boldsymbol{A}_{it}$ represents the ith Eigenvalue at time step t in the ground truth, while $\hat{\boldsymbol{A}}_{it}$ signifies the predicted Eigenvalues. Please take note that further elaboration will be provided in subsequent sections regarding the replacement of $\boldsymbol{A}$ with its normalized counterpart.

3. **Backward Propagation:**
   During backpropagation, the gradients of the loss function are computed with respect to the network's parameters. Starting from the output layer $g$, the gradients are calculated and propagated backward through the network using the chain rule. This process enables the network to determine the contribution of each parameter to the overall loss.

4. **Parameter Update:**
   In this step, the network parameters are adjusted to minimize the loss. This is achieved by employing an optimizer, such as the Adam optimizer [19]. The parameter update involves taking small steps in the opposite direction of the gradients, scaled by a learning rate.

These steps are iteratively repeated (epochs) to determine the optimal network parameters.

## 3.2 Long Short Term Memory

Modeling elastoplastic materials requires the inclusion of a memory component, which is not included within the standard Deep Neural Networks (DNNs). To address this issue, a recurrent neural networks (RNNs) is chosen which uses recursion in the data flow. In addition to receiving the current input $\boldsymbol{X}_t$, RNN neurons also receive the output of the node at time $t-1$, referred to as the hidden state. Unfortunately, simple RNNs suffer from the vanishing gradient problem [20], where the internal state diminishes as the sequence length increases, making it challenging to compute gradients and train the network. As a solution, LSTM blocks have been developed, which incorporate two separate streams known as long-term and short-term memory [21].

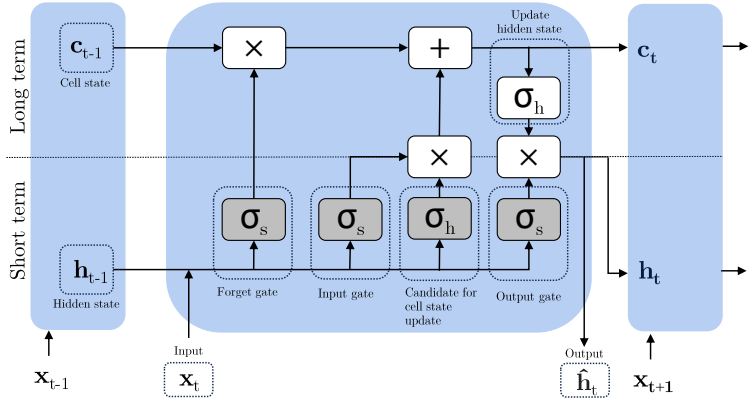



**Fig. 3**: LSTM Cell with the trainable parameters highlighted within the shaded blocks.

Figure 3 illustrates these components, consisting of the cell state $\boldsymbol{c}_t$ for long-term memory and the hidden state $\hat{\boldsymbol{h}}_t$ for short-term memory. The input gate $i_t$, output gate $o_t$, and forget gate $f_t$ are responsible for updating these parameters, as described by the following equations:

$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma_s \left( \boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \hat{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_f \right), \\
\boldsymbol{i}_t &= \sigma_s \left( \boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \hat{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_i \right), \\
\boldsymbol{o}_t &= \sigma_s \left( \boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \hat{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_o \right), \\
\tilde{c}_t &= \sigma_h \left( \boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \hat{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_c \right), \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t, \\
\hat{\boldsymbol{h}}_t &= \boldsymbol{o}_t \odot \sigma_h \left( \boldsymbol{c}_t \right)
\end{aligned}
\tag{28}
$$

In the above equations, the sigmoid activation function is denoted by $\sigma_s$ the hyperbolic tangent function by $\sigma_h$, and $\tilde{\boldsymbol{c}}_t$ represents the activation of the cell state. The matrices $\boldsymbol{W}$ and $\boldsymbol{U}$ contain the weights and recurrent connections, respectively.

The cell state is obtained by element-wise multiplication ($\odot$) between the forget gate, which determines how much of the previous cell state is discarded based on the previous hidden state, and the current input. Subsequently, the input gate $i_t$ controls the amount of new information $\tilde{\boldsymbol{c}}_t$ added to the cell state. For the hidden state, the output gate $o_t$ is multiplied by the hyperbolic tangent activation function applied to the cell state. Note that the initial values are $c_0 = 0$ and $h_0 = 0$.

## 3.3 Multi-Task Learning

Multi-task learning (MTL) is a machine learning subfield where a shared model learns multiple related tasks simultaneously, offering advantages such as improved data efficiency, reduced overfitting, and accelerated learning through auxiliary information [22]. Overfitting is a phenomenon in machine learning where a model becomes excessively specialized in fitting the training data. This involves capturing not only the underlying patterns but also noise or random fluctuations, which can lead to subpar performance when the model encounters new, unseen data. This paper employs hard parameter sharing, depicted in Figure 4, a common approach where hidden layers are shared across tasks with separate output layers, initially introduced by Caruana et al. [23].

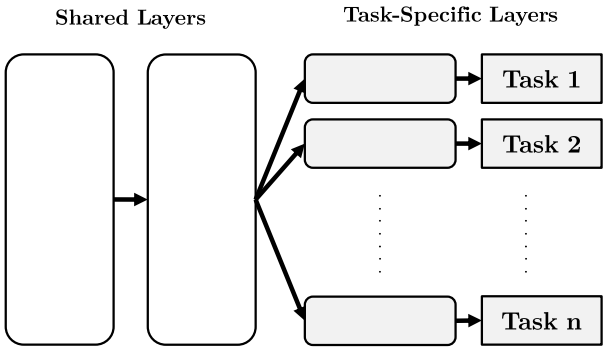



**Fig. 4**: Hard parameter sharing

Hard parameter sharing effectively mitigates the risk of overfitting. Baxter [24] demonstrated that the risk of overfitting shared parameters is significantly lower than that of task-specific parameters. This aligns with the intuitive notion that as more tasks are learned concurrently, the model must find a representation that accommodates all tasks, reducing the risk of overfitting on the original task.

The utilization of hard parameter sharing in the hidden layers offers an additional advantage, particularly when addressing predictions related to material behavior. Specifically, the shared hidden layers are compelled to acquire features that have universal utility across all tasks. This incentivizes the model to discover a comprehensive latent representation that encompasses the common underlying factors and interrelationships among various output variables. To illustrate, in the context of elastoplasticity, these shared layers would assimilate concepts such as stress-strain curves, plastic flow, and hardening, all of which are pertinent to forecasting all field variables.

Concurrently, the distinct task-specific output layers maintain the adaptability necessary for specialization in each variable. For example, layers responsible for predicting plastic strain can concentrate on capturing the nuances of plastic flow rules, while the layers devoted to displacement prediction can excel in representing deformation gradients. This fusion of shared and task-specific parameters empowers the multi-task model to achieve a unified perspective of the overall elastoplastic state while retaining the capacity to optimize for each unique output variable.

# 4 Implementation

The proposed methodology and underlying theoretical framework have been implemented using Python code. Data collection was facilitated by an open-source finite element code with easily interpretable code [16].

## 4.1 Framework

The surrogate model development process, as illustrated in Figure 5, begins with the creation of a mesh and performing finite element (FE) calculations given the boundary values. Multiple FE calculations are performed to construct a diverse dataset encompassing various boundary conditions. After data collection, the full order data $\boldsymbol{S}$ of each individual field variable undergoes dimensionality reduction. Following the dimensionality reduction process, the resulting dataset is partitioned into distinct subsets designated for training and validation. Subsequently, the same process is followed for testing. Additionally, the data are normalized to ensure consistency during training of the neural network.

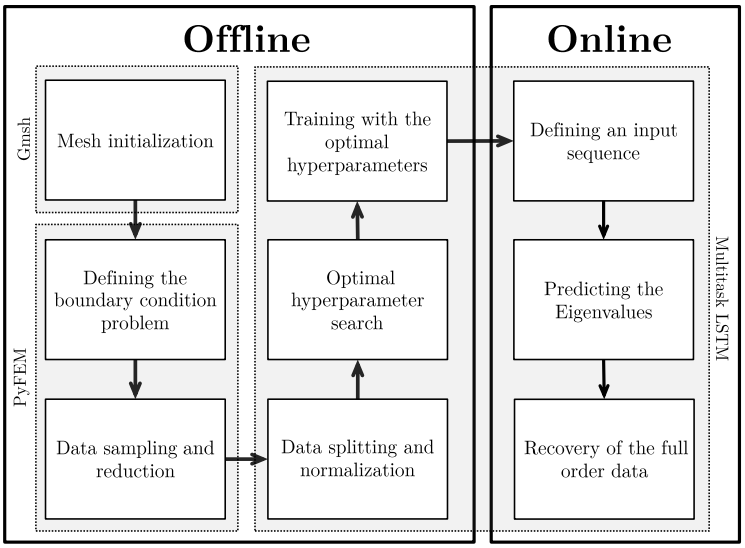



**Fig. 5**: The procedure for creating a surrogate model.

To optimize the neural network's performance, a search is conducted to determine the optimal hyperparameters. This search systematically explores different combinations of hyperparameters with the aim of identifying the set that yields the best performance on the validation set. Once the optimal hyperparameters are determined, the neural network is trained using the compressed and normalized training dataset.

In the online phase, only an input sequence of the boundary conditions needs to be specified to predict the POD data using the trained neural network. This allows for quick and efficient predictions of the reduced-order data based on the input conditions. Finally, the full-order data can be recovered by applying the inverse transformation of the POD technique.

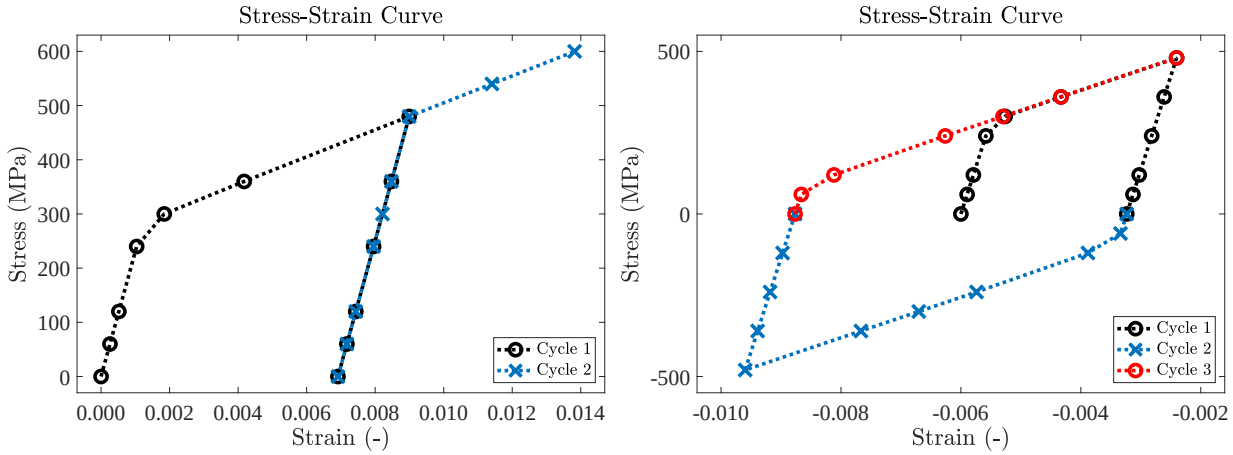
## 4.2 Data Collection

The two-dimensional simulation of the elastoplastic material model utilizes a non-linear solver and small strain elements. It assumes plane stress conditions, where both thickness and stresses in the z direction are considered negligible, and relies on the material properties detailed in Table 1. It's worth mentioning that these parameters were chosen to align with those employed by Im et al. [15]. However, these values do not represent the properties of an actual material.

**Table 1**: Properties of the used elasto-plastic material

| **Property** | **Symbol** | **Value** | **Unit** |
|---|---|---|---|
| Young's Modulus | $E$ | 210 | GPa |
| Poisson's ratio | $v$ | 0.3 | - |
| Initial yield stress | $\sigma_y$ | 250 | MPa |
| Kinematic Hardening Modulus | $H$ | 21 | GPa |
| Isotropic Hardening Scalar | $\gamma$ | 0 | - |

Simulating an uniaxial tensile test in tension produces the behavior shown in Figure 6a. The slope in the elastic region follows the Young's modulus $E$, while the kinematic hardening modulus $H$ describes the gradient in the plastic region. The initial yield stress ($\sigma_y$) defines the boundary between the elastic and plastic ranges.




(a) Tension loading

(b) Cyclic loading

**Fig. 6**: Loading examples

In Figure 6b, cyclic loading is depicted, wherein both uniaxial tension and compression is conducted. The outcome of this test reveals the distinct characteristics of the kinematic hardening.

### 4.2.1 Data Preprocessing

To evaluate the neural network's mapping capabilities, the selection focuses on fundamental variables, each associated with specific dimensions, as outlined in Table 2. The dimensionality is indicative of the number of dimensions per node.

**Table 2**: Used variables

| **Variable** | **Symbol** | **Dimensions ($d$)** | **Unit** |
|---|---|---|---|
| Deformation | $u$ | 2 | mm |
| Von Mises Stress | $\sigma_{\mathrm{vm}}$ | 1 | MPa |
| Equivalent plastic strain | $\bar{\varepsilon}_{\mathrm{pl}}$ | 1 | - |
| Plastic strain tensor | $\varepsilon_{\mathrm{pl}}$ | 6 | - |

For each specific use case, a total of 300 samples ($n_i = 300$) are generated with 10 loading steps ($n_t = 10$). Consequently the following procedure is applied to each variable:

1. **Data reduction:**
   Data reduction is applied to $\boldsymbol{S}$. To minimize information loss, the threshold is set to $e_{\mathrm{RMSE\%}} = 1\%$. This process results in a modal matrix $\boldsymbol{T} \in \mathbb{R}^{r \times n}$ and an Eigenvalue matrix $\boldsymbol{A} \in \mathbb{R}^{r \times m}$. It's worth emphasizing that each variable possesses its own unique value of $r$.

2. **Data sequencing:**
   Dimension $m$ is partitioned into its original number of histories (samples) $n_{\mathrm{i}}$ and $n_{\mathrm{t}}$ loading steps within one history, where $m = n_{\mathrm{i}} \times n_{\mathrm{t}}$. This transformation yields the unnormalized Eigenvalues represented as $\boldsymbol{Y}^* \in \mathbb{R}^{n_i \times n_t \times r}$ and the unnormalized boundary conditions as $\boldsymbol{X}^* \in \mathbb{R}^{n_i \times n_t \times 2 \times d}$. It's important to note that the first loading step in each sample the force magnitude is set to zero.

3. **Data splitting:**
   Among these reduced-order samples, 80% of $n_i$ are assigned for training purposes, while the remaining 20% constitute the validation set. The test set comprises unique scenarios designed to evaluate the neural network's performance under previously unencountered conditions. To mitigate extrapolation errors, the upper and lower bounds of these test samples intentionally adhere to a certain percentage of the values observed in the training set. The test set includes scenarios as detailed in Table 3, along with their corresponding maximum and minimum values.

**Table 3**: Test Set Cases

| | **Loading Case** | **Description** | **Maximum/Minimum [% of max training]** |
|---|---|---|---|
| 1. | Ramp Up | In this scenario, a positively increasing force is steadily applied. | 85% |
| 2. | Bidirectional Cyclic Loading | This case involves incremental variations of the applied force, starting from the minimum and reaching the maximum value of the test set. The sample is then unloaded and reloaded to its minimum value. | 85% |
| 3. | Unidirectional Cyclic Loading | Here, the applied force increases incrementally from the minimum value of the test set to the maximum, after which the sample is unloaded and reloaded to the maximum value. | 85% |
| 4. | Constant load | This scenario maintains a constant force. | 70% |
| 5. | Impulse Load | An alternating force oscillates between its maximum and the minimum values | 70% |

Ensuring alignment between the transformation matrix $\boldsymbol{T}$ of the test set and those of the training and validation sets is vital. To achieve this, the Eigenvalue matrix $\boldsymbol{A}$ of the test set is calculated using $\boldsymbol{T}$ from the training and validation sets according to Equation (20).

4. **Data normalization:**
Data normalization utilizes min-max normalization due to its suitability for uniformly distributed data. The formula for obtaining the normalized input vector $\mathbf{x}'$ is expressed as:

$$\mathbf{X}' = \left[ \frac{\mathbf{x}^{*1} - \boldsymbol{x}^{*1}_{\min}}{\boldsymbol{x}^{*1}_{\max} - \boldsymbol{x}^{*1}_{\min}}, \quad \frac{\mathbf{x}^{*2} - \boldsymbol{x}^{*2}_{\min}}{\boldsymbol{x}^{*2}_{\max} - \boldsymbol{x}^{*2}_{\min}} \right]^T \tag{29}$$

In this context, $\boldsymbol{x}_{\min}$ corresponds to a matrix where all elements are set to the minimum value found in the training dataset, and $\boldsymbol{x}_{\max}$ is a matrix with all elements set to the maximum value from the training set. It's essential to highlight that each element of $\mathbf{X}'$ undergoes independent normalization.
For the output, Z-score standardization is employed due to its Gaussian distribution of . The resulting standardized matrix $\mathbf{Y}'$ is computed as follows:

$$\mathbf{Y}' = \frac{\boldsymbol{Y}^* - \boldsymbol{\mu}_{Y^*}}{\boldsymbol{\sigma}_{Y^*}} \tag{30}$$

In this equation, $\boldsymbol{\mu}_{Y^*}$ signifies the mean matrix, and $\boldsymbol{\sigma}_{Y^*}$ signifies the standard deviation matrix of $\boldsymbol{Y}^*$ within the training dataset. These matrices are broadcasted, meaning they have the same dimensions as $\boldsymbol{Y}^*$, with identical values in all their elements. It is imperative to note that this standardization procedure maintains consistency across all datasets to ensure uniformity.

## 4.3 Model

### 4.3.1 Model Architecture

Unlike the approach used by Im et al. [15], where separate neural networks are used for each variable, a multi-task neural network is utilized. This multi-task network utilizes shared layers followed by task-specific layers, as illustrated in Figure 7.

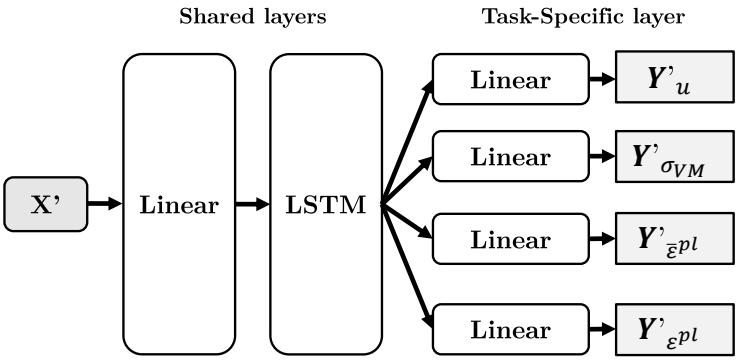



**Fig. 7**: Architecture of the neural network

Following the neural network's prediction of normalized Eigenvalues $\boldsymbol{Y}'$, they undergo a denormalization process to $\boldsymbol{Y}^*$ and are converted into $\boldsymbol{S}^{\text{rec}}$ through the utilization of Equation (21). The determination of the network's architecture hyperparameters is elucidated in the subsequent section. These hyperparameters encompass the hidden state size within the LSTM layer(s), and the number of LSTM layers.

### 4.3.2 Hyperparameter Tuning

Hyperparameter tuning is a critical aspect of optimizing neural network training, as selecting appropriate hyperparameters greatly impacts the model's performance [25]. While Im et al. [15] used a grid search to find suitable hyperparameter combinations, a different approach is adopted. Specifically, the capabilities of an external Python library called Optuna [26] are used, which utilizes a Tree Parzen Estimator (TPE), a Bayesian optimization algorithm [27].

To initiate the hyperparameter search, specific value ranges are defined for each hyperparameter, as detailed in Table 4. The hyperparameters encompass the batch size, the hidden state size of the LSTM block, the learning rate, and the weight decay. The number of training rounds (Epochs) is set to 5000.
Table 4 presents the hyperparameters and their respective ranges. The selection of these ranges is the result of broader initial ranges, from which it was deduced that the listed regions represent the optimal ranges for both cases.

**Table 4**: Hyperparameters

| **Hyperparameter** | **Range** |
|---|---|
| Batch Size | [1, 128] |
| LSTM Hidden State Size | [r, 100] |
| Learning Rate | $[1 \times 10^{-3}, 1 \times 10^{-1}]$ |
| Number of LSTM Layers | [1,5] |
| Weight Decay | $[1 \times 10^{-7}, 1 \times 10^{-4}]$ |

During the optimization process, uniform distributions are constructed from the specified ranges, serving as the priors, which get updated as the search progresses. Thereafter the hyperparameter combinations are iteratively sampled and evaluated to their performance based on the mean RMSE of the validation data. The network is trained three times for each combination to obtain a robust evaluation. From the mean of these evaluations, two distributions are derived: a good distribution ($l(x)$) containing the top 20% of samples and a bad distribution ($g(x)$). The goal is to maximize the ratio $\frac{l(x)}{g(x)}$ to determine the next hyperparameter to evaluate. This iterative process continues until the local optimal hyperparameter combination is identified, optimizing the performance of the model for the given task.

### 4.3.3 Model Evaluation

To assess the performance, the key metrics used are Mean Absolute Error (MAE) and the Weighted coefficient of determination ($R^2$). The MAE measures the absolute deviation between the predicted reduced-order matrix $\hat{\boldsymbol{S}}^{\text{rec}}$ and its true counterpart $\boldsymbol{S}^{\text{rec}}$,quantifying the error as shown below:

$$e_{\text{MAE}} = \frac{1}{m \times n} \sum_{t=1}^{m} \sum_{i=1}^{n} |S_{it}^{\text{rec}} - \hat{S}_{it}^{\text{rec}}| \tag{31}$$

While $e_{\text{MAE}}$ provides insight into the absolute difference between prediction and ground truth, it's equally important to consider the error in percentage terms. To calculate percentage error, the absolute error is weighted by the maximum value of the ground truth for a specific sample, as expressed in Equation (32).

$$e_{\text{MAE\%}} = \frac{e_{\text{MAE}}}{S^{\text{rec}}_{\text{max}}} \tag{32}$$

Here, $S^{\text{rec}}_{\text{max}}$ denotes the maximum value of $\boldsymbol{S}^{\text{rec}}$. However, it's important to acknowledge cases where the maximum value for a specific variable remains zero throughout all time steps of a sample. For instance, this occurs with plastic strain when the yield stress is not exceeded, resulting in division by zero when calculating percentage error. To mitigate this issue, we adopt a strategy of using the maximum value across the entire dataset, ensuring consistency by setting $S^{\text{rec}}_{\text{max}}$ to the maximum of the training set. It's worth mentioning that this approach differs slightly from the method employed by Im et al. [15], as they determine the percentage with use of the maximum within the sample under evaluation.

It's worth noting that both $e_{\text{MAE}}$ and $e_{\text{MAE\%}}$ are computed with respect to the recovered data, focusing on errors within the physical space. This may lead to skewed results towards zero, especially when the initial data predominantly consists of zero values, such as in use case 2.

In situations involving sparse system matrices, it proves advantageous to evaluate the quality of predictions against the ground truth using the coefficient of determination, denoted as $\text{R}^2$. This metric aids in assessing the neural network's ability to elucidate the variance in the predicted Eigenvalues. The equation for the Weighted $R^2$ is expressed as:

$$\text{Weighted } R^2 = \frac{\sum_{i=1}^{r} |A_{i,\max} - A_{i,\min}|}{\sum_{i=1}^{r} |A_{i,\max} - A_{i,\min}|} \left(1 - \frac{\sum_{t=1}^{m} (\hat{A}_{it} - \bar{A}_i)^2}{\sum_{t=1}^{m} (A_{it} - \bar{A}_i)^2}\right) \tag{33}$$

In this equation, $\bar{A}_i \in \mathbb{R}^r$ represents the mean of the true Eigenvalues for each mode, while $A_{i,\max}$ and $A_{i,\min}$ denote the maximum and minimum values of the $i$th row, respectively. $A_{i,\max}$ and $A_{i,\min}$ represent the overall maximum and minimum values of $\boldsymbol{A}$. Given the greater significance of the first Eigenvalue in accurately representing the variable, the weighting of $R^2$ is adjusted accordingly. The process to obtain $\boldsymbol{A}$ involves unnormalizing $Y'$ and subsequently resizing the matrix to its original dimensions before sequencing.

These metrics are computed on both the validation set, and the test set. While the validation performance guides the hyperparameter optimization, the test performance provides an unbiased estimate of the model's ability to generalize to new data.

Reporting both $e_{\text{MAE\%}}$ and the Weighted $R^2$ on the validation and test sets provides a comprehensive understanding of model performance. The goal is to attain low $e_{\text{MAE\%}}$ and high $R^2$ (towards 1) values across both sets, which indicate accurate predictions and strong generalization.

# 5 Model Demonstrations

The proposed LSTM multi-task framework's performance was evaluated on two 2D use cases: a table model and a cantilever beam model. Results for each case showcase the model's ability to predict key field variables accurately under special loading cases such as cyclic loading. The framework demonstrates competence in capturing nonlinear elastoplastic behavior and making precise surrogate predictions in real-time.

## 5.1 Use Case 1

To evaluate the proposed Multi-Task LSTM framework's effectiveness, an initial assessment was conducted using a 2D table model. The model, originally utilized by Im et al. [15] to showcase the LSTM network's performance, was replicated to create a benchmark for comparative evaluation. This duplication serves as a validation of the multi-task approach's effectiveness. The FEM model comprises of 800 discrete 2D quad elements, and 966 nodes. For a comprehensive understanding of the geometry and configuration of the table, please refer to Figure 8.

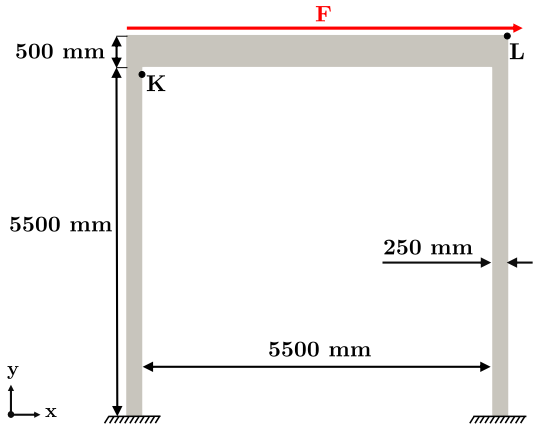



**Fig. 8**: Table model

By systematically applying a varied range of distributed forces over 10 time steps on the model's upper surface, the necessary datasets were generated as outlined. These forces are represented as $\boldsymbol{X} = \boldsymbol{F} \in [-21\ \text{MN}, 21\ \text{MN}]$ when converted into force from the pressure.

### 5.1.1 Data Reduction

The total dimension of the data varies according to the field of the specific variable. Table 5 presents these numbers, both before and after applying the SVD reduction method with an $e_{\text{RMSE}\%}$ threshold of 1%.

**Table 5**: Data dimensionality

| | **Full-order** | **Reduced-order** |
|---|---|---|
| $u$ | 1,932 | 1 |
| $\sigma_{\text{vm}}$ | 966 | 8 |
| $\bar{\varepsilon}_{\text{pl}}$ | 966 | 4 |
| $\varepsilon_{\text{pl}}$ | 63,756 | 3 |

This reduction highlights the substantial data reduction accomplished while preserving crucial information, thereby facilitating the neural network in predicting the correct output without an excessive number of parameters to train. In comparison to Im et al. [15], the reduced dimensions are approximately at the same order of magnitude, though not identical. This variation might arise from differences in mesh structure, the computation of $e_{\text{RMSE\%}}$, or variations in the FEM calculation configurations. Figure 9 illustrates the essential modes of equivalent plastic strain. This visualization prominently showcases localized positions of plastic strain unique to each mode. The other variables exhibit similar responses, except for von Mises stress. These stress contours remain quite consistent across different modes, displaying minor deviations. To accommodate these slight differences, a larger number of dimensions is required to accurately describe each state.

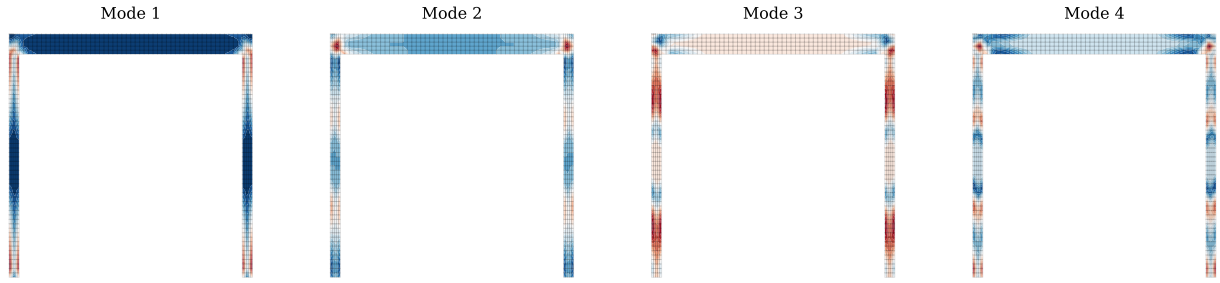



**Fig. 9**: The modes of the equivalent plastic strain.

### 5.1.2 Results

Training the neural network to predict the Eigenvalues matrix $A$ is done during the hyperparameter search. The optimal hyperparameters to predict these values for this specific dataset are listed in Table 6 With the selected hyperparameters, the network consists of only 8,058 trainable parameters (weights and biases) of which approximately 8 % is dedicated to transform the hidden latent space into the variable specific Eigenvalues. Using the metrics described yielded the results as listed in Table 7.

**Table 6**: Hyperparameters of the neural network

| **Hyperparameter** | **Value** |
|---|---|
| Batch Size | 32 |
| LSTM Hidden State Size | 41 |
| Learning Rate | 0.045 |
| Number of LSTM Layers | 1 |
| Epochs | 5000 |
| Weight Decay | $5.4 \times 10^{-4}$ |

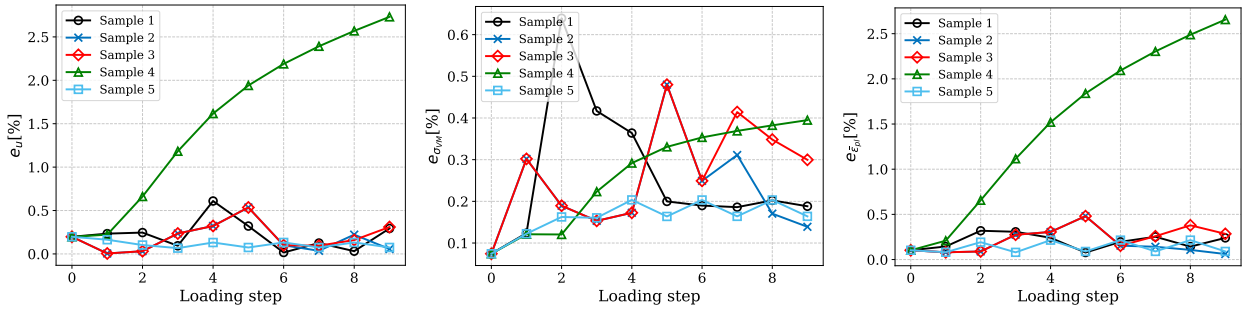
**Table 7**: Model evaluation

| | $e_{\text{MAE\%}}$ | | Weighted $R^2$ | |
|---|---|---|---|---|
| | **Validation** | **Test** | **Validation** | **Test** |
| $u$ | 0.21% | 0.46% | 1.00 | 1.00 |
| $\sigma_{\text{vm}}$ | 0.26% | 0.24% | 0.95 | 0.97 |
| $\bar{\varepsilon}_{\text{pl}}$ | 0.23% | 0.45% | 0.98 | 0.96 |
| $\varepsilon_{\text{pl}}$ | 0.07% | 0.14% | 0.99 | 0.99 |

The findings underscore the effectiveness of the proposed LSTM multi-task framework in accurately predicting the field variables, such as displacement and equivalent plastic strain. The model exhibits relatively low $e_{\text{MAE\%}}$ and Weighted ($R^2$) values across both the validation and test datasets, indicating its capacity to faithfully capture the nonlinear elastoplastic behavior and generalize to new data.

The similarity in error magnitudes between the validation and test datasets signifies the model's robust generalization to previously unseen data points. Even for more intricate variables like the plastic strain tensor, the multi-task framework achieves promising MAE percentages of 0.07% on validation and 0.14% on the test dataset.

It's noteworthy that these results align with those reported by Im et al. [15], who observed $e_{\mathrm{MAE}\%}$ values ranging from 0.88% to 1.09% in their deformation predictions for their test dataset. The high $R^2$ values can be attributed to the standardized output, implying that the neural network primarily needs to learn deviations in relation to the input. When examining the progression of error across the time steps in the test set samples, one can discern both the limitations and strengths of the trained network. Figure 10 presents the errors for the key field variables.




(a) Displacement error (b) Von Mises stress error (c) Equivalent plastic strain error

**Fig. 10**: Error trend over time for the test cases.

It is evident that for sample 4, where the force remains constant, the error consistently increases. This behavior arises from the inherent nature of the LSTM, which relies on previous states to predict subsequent ones. Additionally, it is noteworthy that the error observed in the case of von Mises stress is notably smaller when compared to the other test samples.

Furthermore, in the case of sample 5, a noticeable bias towards the force's sign is observed. Time steps characterized by a negative force tend to exhibit higher error, as evident from the fluctuations. Given the pivotal role of kinematic hardening in material behavior and its history dependency, particular attention will be focused on a cyclic loading case (sample 3). Figure 11 illustrates the applied force at each loading step for sample 3, with a focus on three specific reference steps during unloading.

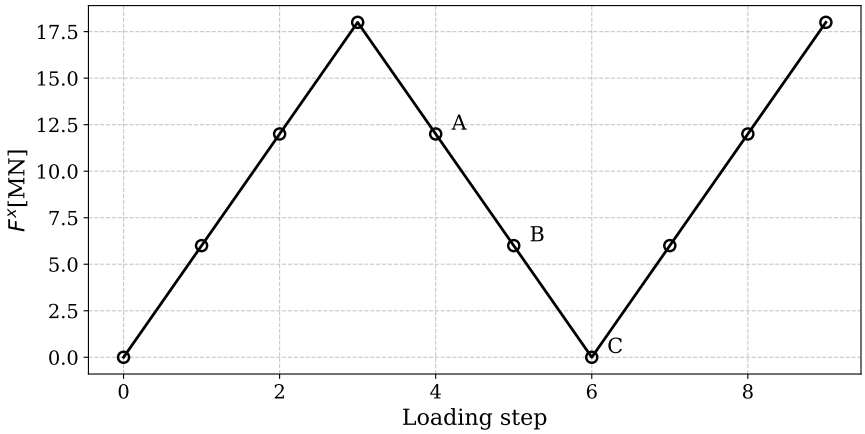
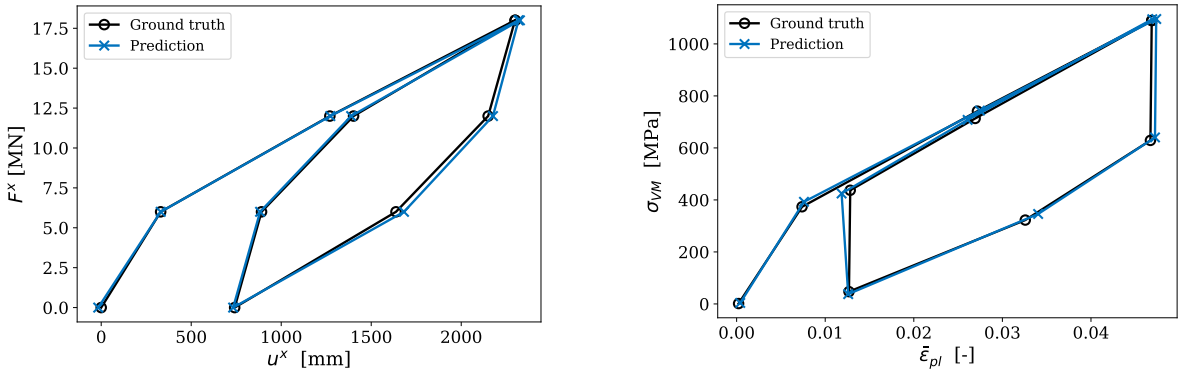



**Fig. 11**: Sample 3: Force history

Measurement of strain at the point where equivalent plastic strain is concentrated (point K) as depicted in Figure 12, illustrates the model's capability to capture the essence of material behavior.




(a) Force-strain curve at B

(b) Stress-strain curve at C

**Fig. 12**: Sample 3: Material response

When observing the resulting contour plot of the equivalent plastic strain as depicted in Figure 13, the error is primarily visible in the legs, where most of the variation is concentrated, as seen in the modes. Therefore, it is imperative to accurately predict every Eigenvalue to minimize the error in this critical area.

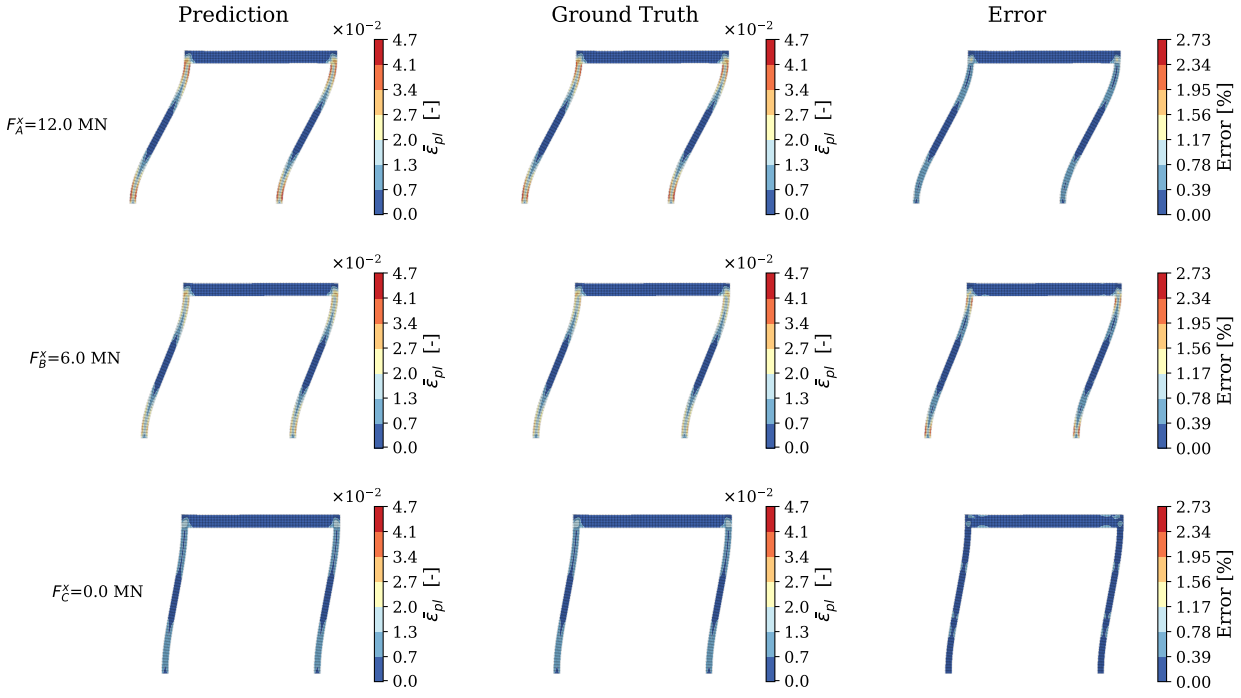
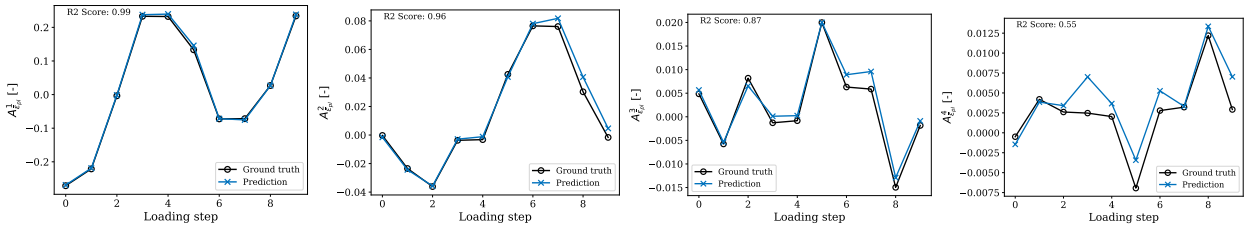



**Fig. 13**: Sample 3: Equivalent plastic strain at each reference point.

Examining the predicted Eigenvalues in Figure 14, it becomes apparent that the neural network focuses on getting the first Eigenvalue correct, as it holds significant importance in reducing the overall error.




(a) First Eigenvalue (b) Second Eigenvalue (c) Third Eigenvalue (d) Fourth Eigenvalue

**Fig. 14**: Sample 3: Equivalent plastic strain Eigenvalues over time.

The prediction time for the machine learning model and the computation time for the FEM are detailed in Table 8, along with the hardware configurations used to obtain these results. It's important to note that the machine learning model leverages a GPU for parallel computations, while the FEM relies on a CPU, as it operates as a serial process. Notably, the machine learning model demonstrates a substantial speedup of approximately 640,000. While alternative FEM approaches may yield faster computations, the achieved speedup remains noteworthy.

**Table 8**: Comparison of computation times

| **Method** | **Average time per sample** | **Hardware** |
|---|---|---|
| Machine Learning model | 2.5 $\times 10^{-4}$ seconds | NVIDIA T4 GPU |
| Conventional FEM | 1.6 $\times 10^{2}$ seconds | Intel Core i7-7700HQ CPU |

The quantitative results presented in this section affirm the efficacy of the proposed approach when applied to the table model, establishing a robust foundation for further investigation. The LSTM model's adeptness in capturing the nonlinear and path-dependent behavior using reduced-order data is evident. With these promising findings, the next logical progression is to extend this methodology to the analysis of a cantilever beam, allowing for a comprehensive exploration of its capabilities and insights into broader applications.

### 5.1.3 Multi-Task Versus Single-Task

The multi-task LSTM model offers clear advantages over training individual models for each output variable. By learning shared representations across tasks, this approach enhances generalization across all output variables, resulting in reduced error rates when compared to single-task models. The differences between these two model types are presented in Table 15, where positive values signify improvements achieved by the multi-task model over the single-task model.

**Table 9**: Difference single versus multi-task

| | $e_{\mathbf{MAE\%}}$ | | **Weighted $R^2$** | |
|---|---|---|---|---|
| | **Validation** | **Test** | **Validation** | **Test** |
| $u$ | +0.12% | +0.24% | +0.01 | +0.01 |
| $\sigma_{\text{vm}}$ | +0.03 % | +0.11% | 0.00 | +0.05 |
| $\bar{\varepsilon}_{\text{pl}}$ | +0.09% | +0.10% | 0.00 | 0.00 |
| $\varepsilon_{\text{pl}}$ | +0.01% | 0.00% | 0.00 | 0.00 |

It is essential to note that although the differences may appear small, the multi-task model consistently outperforms the single-task model for each variable, particularly in the case of $e_{\mathrm{MAE\%}}$ on the test set, which exhibits the most substantial improvement. This improvement is primarily observed for displacement ($u$), as this variable consists of a single Eigenvalue ($r = 1$). Consequently, the single-task model is more prone to overfitting, while the multi-task model excels in generalization due to its shared layers, which mitigate overfitting tendencies.

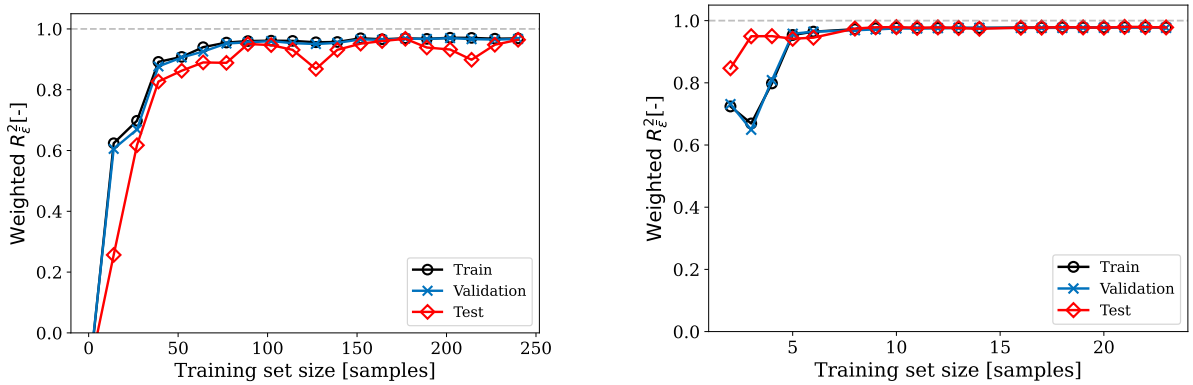
An additional benefit emerges when incorporating additional variables like the total equivalent strain ($\bar{\varepsilon}$). This inclusion prompts improvements in the shared layers, leading to better predictions for other closely related variables. When adding $\bar{\varepsilon}$ to the output variables, the differences detailed in Table 10 are observed.

**Table 10**: Enhancements with added input $\bar{\varepsilon}$.

| | $e_{\mathrm{MAE\%}}$ | | Weighted $R^2$ | |
|---|---|---|---|---|
| | **Validation** | **Test** | **Validation** | **Test** |
| $u$ | +0.01% | +0.11% | 0.00 | 0.00 |
| $\sigma_{\mathrm{vm}}$ | -0.02% | -0.05% | -0.01 | -0.02 |
| $\bar{\varepsilon}_{\mathrm{pl}}$ | +0.01% | +0.10% | 0.00 | +0.01 |
| $\varepsilon_{\mathrm{pl}}$ | 0.00% | +0.02% | 0.00 | 0.00 |

Improvements are noticeable in most variables, except von Mises stress. This divergence is due to lower correlation with equivalent strain and the need to predict more Eigenvalues for the von Mises stress ($r = 8$). Consequently, including equivalent strain slightly reduces performance. Conversely, adding variables like the stress tensor, which correlates more strongly with von Mises stress, can counteract this effect. Adjusting loss weights for specific variables can also improve results, depending on the emphasis desired for specific variables.

A pre-trained multi-task model can effectively train additional variables with fewer samples due to its comprehensive understanding. For instance, with equivalent strain ($\bar{\varepsilon}$) as an example, only 7 samples achieve convergence. In contrast, a single-task model needs around 100 samples to reach a similar performance level, as seen in Figure 15. Notably, the single-task model exhibits fluctuations around 140 and 210 samples due to overfitting, resulting in inferior test set results compared to other datasets.




(a) Single-task model

(b) Multi-task model

**Fig. 15**: Data sensitivity on the Weighted $R^2$ with additional input $\bar{\varepsilon}$.

In addition to significantly reducing the training time by a factor of approximately four times the number of trained variables (4 in this case), there is also a notable decrease in prediction time, which drops from 0.70 ms to 0.25 ms when handling four variables. This reduction in prediction time can be attributed to the adoption of a multi-task approach that employs a single, streamlined model for predictions. This approach enables faster real-time predictions. While this speed improvement may seem relatively modest, approximately three times faster, it becomes particularly advantageous for more complicated problems with a larger number of inputs and outputs.

## 5.2 Use case 2

In this section, we assess the performance of the proposed LSTM multi-task framework using a 2D cantilever problem. The FEM model consists of 980 discrete quadrilateral elements and 1128 nodes. The datasets for the field variables were generated by systematically applying a varied range of concentrated point forces over a span of 10 time steps while simultaneously adjusting the position at the upper end of the beam. The input range can be represented as follows:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{P} \end{bmatrix} = \begin{bmatrix} [-6\,\mathrm{N}, 6\,\mathrm{N}] \\ [5\,\mathrm{mm}, 10\,\mathrm{mm}] \end{bmatrix}$$

For the test set, all positions are maintained at 7.5 mm, except for the constant force load case (sample 4), where the position gradually increases from the middle to the end of the beam in equal increments. This approach allows for the evaluation of the test set with minimal impact on the additional input parameter while still making its influence evident in one specific sample.

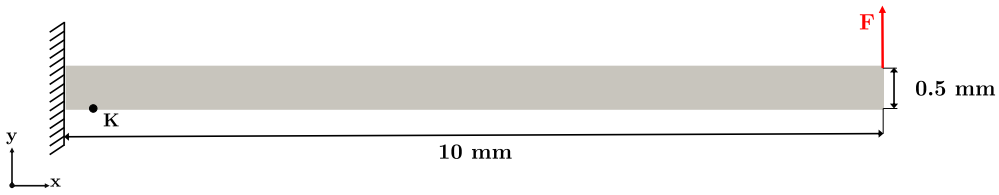



**Fig. 16**: Beam model

### 5.2.1 Data Reduction

After data collection, POD was applied to reduce the dimensionality of the data for each field variable. The dimensions before and after reduction are provided in Table 11.

**Table 11**: Dimensions before and after data reduction.

| Variable | Full-order | Reduced-order |
|---|---|---|
| $u$ | 2,256 | 2 |
| $\sigma_{\mathrm{vm}}$ | 1,128 | 19 |
| $\bar{\varepsilon}_{\mathrm{pl}}$ | 1,128 | 7 |
| $\varepsilon_{\mathrm{pl}}$ | 6,768 | 4 |

In contrast to the table, the variables need to be represented with a greater number of modes. This arises from the subtle distinctions within the modes, primarily attributable to the increased variability in locality

due to the shifting force positions, resulting in a broader range of loaded regions. For the equivalent plastic strain, the first modes are depicted in Figure 17


**Fig. 17**: The first 4 modes of the equivalent plastic strain.

### 5.2.2 Results

The optimal hyperparameters obtained via Bayesian optimization are displayed in Table 12, resulting in a total of 48,608 trainable parameters. The results, as outlined in Table 13, reveal a performance decrease in both $e_{\mathrm{MAE\%}}$ and $R^2$ for the validation compared to the first use case. This can be attributed to the increased complexity of the problem, which necessitates the mapping of both position and magnitude to standardized eigenvalues and the number of modes.

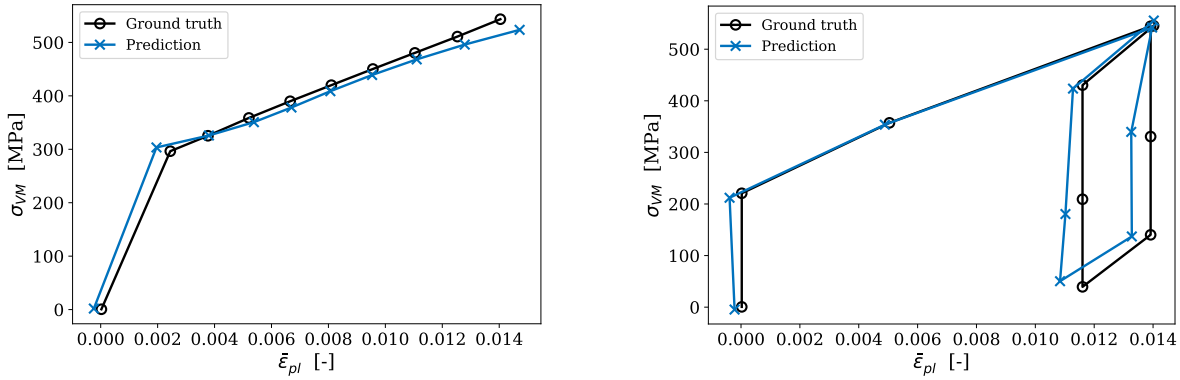
**Table 12**: Hyperparameters of the neural network

| **Hyperparameter** | **Value** |
|---|---|
| Batch Size | 60 |
| LSTM Hidden State Size | 61 |
| Learning Rate | 0.016 |
| Number of LSTM Layers | 2 |
| Epochs | 1000 |
| Weight Decay | $1.7 \times 10^{-5}$ |

**Table 13**: Model evaluation

| | $e_{\mathrm{MAE\%}}$ | | Weighted $R^2$ | |
|---|---|---|---|---|
| | **Validation** | **Test** | **Validation** | **Test** |
| $u$ | 0.21% | 0.15% | 1 | 0.99 |
| $\sigma_{\mathrm{vm}}$ | 0.58 % | 0.48% | 0.91 | 0.91 |
| $\bar{\varepsilon}_{\mathrm{pl}}$ | 0.37% | 0.27% | 0.94 | 0.93 |
| $\varepsilon_{\mathrm{pl}}$ | 0.12% | 0.08% | 0.98 | 0.98 |

Analyzing the stress-strain curve of the 4th sample with varying positions in Figure 18a exposes the neural network's challenges in predicting plastic strain, particularly when dealing with scenarios close to the boundaries of the training data range. In the case of unidirectional cyclic loading (Figure 18b), this stress-strain curve aligns better with the ground truth, as interpolation is more straightforward for these input conditions. However, during the initial loading step and subsequent unloading and loading, this sample exhibits some deviation from the ground truth.

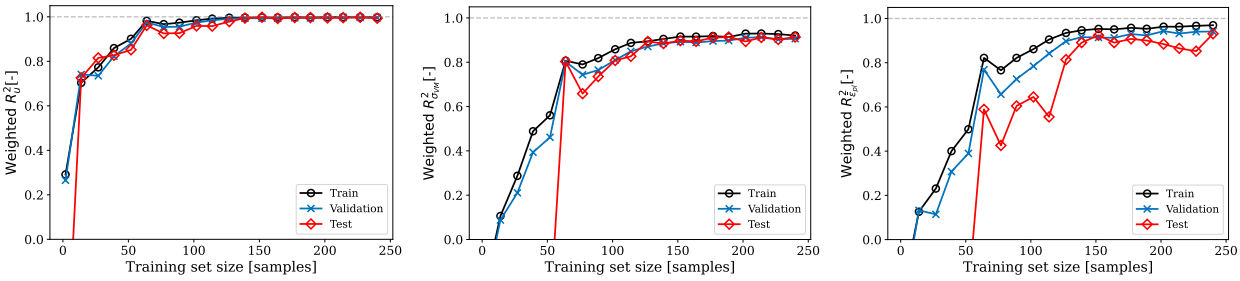



(a) Sample 4: Constant load & varying position

(b) Sample 3: Unidirectional cyclic loading

**Fig. 18**: Stress-strain curve at K.

When reducing the number of training set samples, as illustrated in Figure 19, the Weighted $R_2$ reveals that fewer samples are required for convergence of $R^2$. Displacement, being minimally influenced by its history, necessitates fewer samples, approximately 100, to achieve an $R^2$ close to 1 on the test set. On the other hand, equivalent plastic strain, which is more history-dependent, demands a larger sample size, around 150. However still some fluctuations on the test set can be seen, which is the result of overfitting. The von Mises tress falls in between, with approximately 75 samples needed for convergence. Notably, von Mises stress does not seem to converge to 1, suggesting that the neural network struggles to capture the behavior of the von Mises stress Eigenvalues. This may arise from the complexity of the data compared to the neural network's capacity to capture it. Increasing the number of trainable parameters could be considered as a potential solution.




(a) Displacement (b) Von Mises stress (c) Equivalent Strain

**Fig. 19**: Training data sensitivity on the Weighted $R^2$.

Table 14 shows the ML prediction and FEM computation times. The ML model achieves a significant speedup of around 700,000x.

**Table 14**: Comparison of Computation Times

| **Method** | **Average time per sample** |
|---|---|
| Machine Learning model | $2.8 \times 10^{-4}$ seconds |
| Conventional FEM | $2.0 \times 10^{2}$ seconds |

In the offline phase, generating both the validation and training sets demands approximately 16 hours. However, reducing this extensive dataset to a manageable size is a swift process, taking merely 5 seconds. While data generation is time-consuming, the subsequent training process is highly efficient, completing in just 120 seconds.

### 5.2.3 Multi-Task Versus Single-Task

Examining the distinctions between the multi-task model and the single-task model is detailed in Table 15.

**Table 15**: Difference single versus multi-task.

| | $e_{\mathbf{MAE\%}}$ | | **Weighted $R^2$** | |
|---|---|---|---|---|
| | **Validation** | **Test** | **Validation** | **Test** |
| $u$ | +0.03% | +0.02% | 0.00 | +0.01 |
| $\sigma_{\mathrm{vm}}$ | +0.07 % | +0.13% | 0.00 | +0.02 |
| $\bar{\varepsilon}_{\mathrm{pl}}$ | +0.05% | +0.07% | +0.01 | +0.01 |
| $\varepsilon_{\mathrm{pl}}$ | +0.01% | 0.00% | 0.00 | 0.00 |

Here positive values signify improvements achieved by the multi-task model over the single-task model.

Although the distinctions between both models are slight, it's essential to highlight that the multi-task model demonstrates particular proficiency in capturing the data's variability on the test set, as reflected in the results related to the Weighted $R^2$.

Incorporating extra variables, such as total equivalent strain ($\bar{\varepsilon}$), yields additional advantages. When $\bar{\varepsilon}$ is included in the output variables, similar enhancements to those in Table 16 are evident.

**Table 16**: Enhancements with added input $\bar{\varepsilon}$.

| | $e_{\mathbf{MAE\%}}$ | | **Weighted $R^2$** | |
|---|---|---|---|---|
| | **Validation** | **Test** | **Validation** | **Test** |
| $u$ | +0.06% | +0.04% | 0.00 | +0.01 |
| $\sigma_{\mathrm{vm}}$ | +0.07% | +0.11% | 0.00 | +0.02 |
| $\bar{\varepsilon}_{\mathrm{pl}}$ | +0.08% | +0.10% | +0.01 | +0.01 |
| $\varepsilon_{\mathrm{pl}}$ | 0.00% | 0.00% | 0.00 | 0.00 |

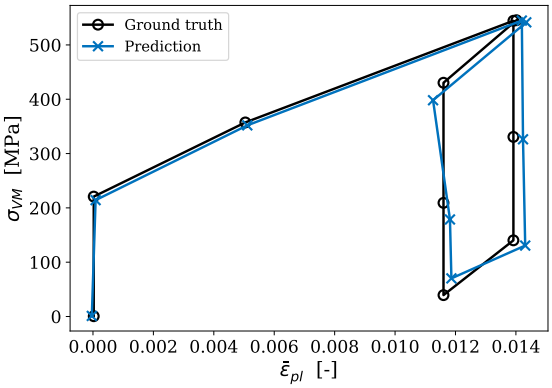
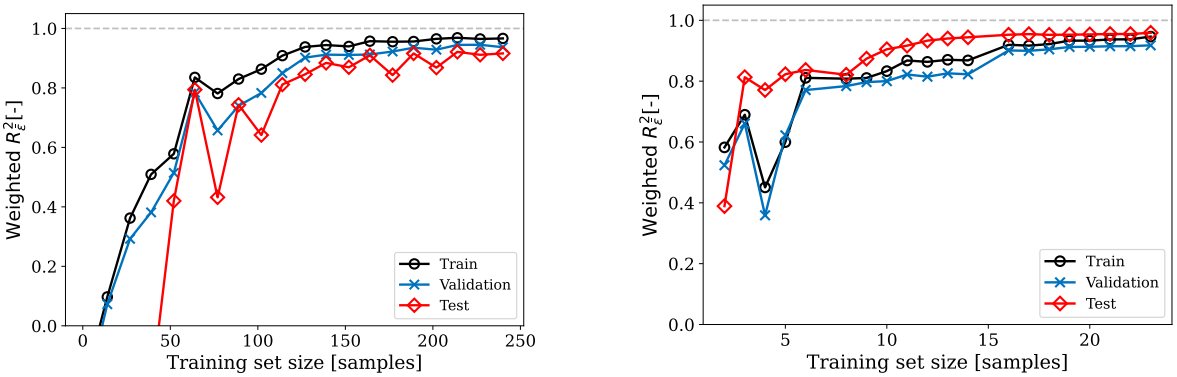



**Fig. 20**: Sample 3: Stress-strain curve with $\bar{\varepsilon}$.

Significantly, the test set shows enhancements in $\bar{\varepsilon}_{\mathrm{pl}}$ with the incorporation of $\bar{\varepsilon}$. This improvement is also clearly visible in the stress-strain curve as previously depicted in Figure 18b, and the updated Figure 20. Particularly noteworthy are the improvements observed during the initial loading step, where elasticity plays a crucial role, and during the unloading phase. In these instances, plastic deformation remains constant, resulting in a vertical line in the stress-strain curve.

Moreover, a pre-trained multi-task model can be leveraged for training additional variables. For instance, in the case of equivalent strain ($\bar{\varepsilon}$), approximately 17 samples are sufficient for convergence, compared to the approximately 150 samples required for the same level of convergence, as shown in Figure 21.




(a) Single-task $R^2$

(b) Multi-task $R^2$

**Fig. 21**: Data sensitivity with additional input $\bar{\varepsilon}$.

# 6 Conclusion And Outlook

This study introduced a novel LSTM multi-task framework for surrogate modeling of elastoplastic deformation problems. The approach combines proper orthogonal decomposition, recurrent neural networks, and multi-task learning to achieve efficient yet accurate real-time predictions. The methodology was validated on two 2D examples.

The results demonstrated the proposed model's effectiveness in predicting nonlinear path-dependent behavior under unseen test samples including cyclic loading cases. For both use cases, the model attained low error rates ($< 0.6\%$) and high Weighted R2 values ($> 0.90$), showcasing its capability to capture intricate material phenomena like the Bauschinger effect and making a prediction in a small amount of time ($< 0.3$ ms).

The multi-task architecture demonstrated clear advantages over individual models. It achieved enhanced generalization and more compact training. The shared layers also enable improved performance when adding new output variables. For instance, incorporating total strain further optimized the shared representation, leading to reduced errors up to 0.13% MAE during testing. Overall, multi-task learning provides a more unified perspective of the system's state while maintaining flexibility. This represents a promising direction for surrogate modeling of complex physics problems.

Nevertheless, certain limitations were observed in the model's performance. Notably, the model encountered challenges in extrapolation, particularly in proximity to the boundaries of the training data range. This limitation became evident in scenarios such as the constant load test on the beam, where the load position varied. Moreover, in cases where the material remained in the plastic regime, the model exhibited increasingly higher errors due to its reliance on previous states. Additionally, it's important to acknowledge that the current model is restricted to a specific geometry and a single material model.

To mitigate these limitations, several strategies can be considered. Firstly, the utilization of graph neural networks or the adoption of non-geometry-specific reduced data techniques could help address issues associated with geometry specificity. Lastly, exploring the integration of different material models as additional input features could enhance the model's applicability across various contexts. Overall, this study demonstrates the potential of the proposed framework as an efficient surrogate modeling technique for nonlinear path-dependent analysis. Key recommendations for future work include:

- Training with various nonlinear material models, such as hyperelasticity, for the purpose of validating flexibility.
- Progressing to 3D problems to enhance applicability to real-world scenarios.
- Implementing strategies to accommodate for different geometries.

The proposed methodology shows promise in accelerating simulations for complex interventions while retaining accuracy. This study lays a robust foundation for applying the framework to patient-specific biomechanical modeling problems, paving the way for significant advancements in surgical assistance and planning.

# References


[1] Legrand, J., Niu, K., Qian, Z., Denis, K., Vander Poorten, V., Van Gerven, L., Vander Poorten, E.: A Method Based on 3D Shape Analysis Towards the Design of Flexible Instruments for Endoscopic Maxillary Sinus Surgery. Annals of Biomedical Engineering **49**(6), 1534–1550 (2021) https://doi.org/10.1007/s10439-020-02700-z

[2] Okereke, M., Keates, S.: Computational Mechanics and the Finite Element Method. In: Okereke, M., Keates, S. (eds.) Finite Element Applications: A Practical Guide to the FEM Process, pp. 3–25. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67125-31

[3] Marinkovic, D., Zehn, M.: Survey of finite element method-based real-time simulations. MDPI AG (2019). https://doi.org/10.3390/app9142775

[4] Morooka, K., Chen, X., Kurazume, R., Uchida, S., Hara, K., Iwashita, Y., Hashizume, M.: Real-Time Nonlinear FEM with Neural Network for Simulating Soft Organ Model Deformation. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2008, 742–749 (2008) https://doi.org/10.1007/978-3-540-85990-1-89

[5] Sanghi, S., Hasan, N.: Proper orthogonal decomposition and its applications. Asia-Pacific Journal of Chemical Engineering **6**(1), 120–128 (2011) https://doi.org/10.1002/apj.481

[6] Soldner, D., Brands, B., Zabihyan, R., Steinmann, P., Mergheim, J.: A numerical study of different projection-based model reduction techniques applied to computational homogenisation. Computational Mechanics **60**(4), 613–625 (2017) https://doi.org/10.1007/s00466-017-1428-x

[7] Bond, B.N., Daniel, L.: A piecewise-linear moment matching Approach to parameterized model order reduction for highly nonlinear systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **26**(12), 2116–2129 (2007) https://doi.org/10.1109/TCAD.2007.907258

[8] Rewieński, M., White, J.: A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems vol. 22, pp. 155–170 (2003). https://doi.org/10.1109/TCAD.2002.806601

[9] Cho, H., Shin, S.J., Kim, H., Cho, M.: Enhanced model-order reduction approach via online adaptation for parametrized nonlinear structural problems. Computational Mechanics **65**(2), 331–353 (2020) https://doi.org/10.1007/s00466-019-01771-7

[10] Lopez-Pacheco, M., Yu, W.: Complex Valued Deep Neural Networks for Nonlinear System Modeling. Neural Processing Letters **54**(1), 559–580 (2022) https://doi.org/10.1007/s11063-021-10644-1

[11] Mendizabal, A.: Machine Learning meets real-time Numerical Simulation - Application to surgical training preoperative planning and surgical assistance. Technical report, Université de Strasbourg, Strasbourg (2020). https://hal.science/tel-03116502v1/document

[12] Odot, A., Haferssas, R., Cotin, S., Cotin, S.: DeepPhysics: a physics aware deep learning framework for real-time simulation. International Journal for Numerical Methods in Engineering **123**(10) (2021)

https://doi.org/10.1002/nme.6943

[13] El Said, B.: Predicting the non-linear response of composite materials using deep recurrent convolutional neural networks. International Journal of Solids and Structures **276** (2023) https://doi.org/10.1016/j.ijsolstr.2023.112334

[14] Brunton, S.L.: Applying machine learning to study fluid mechanics. Acta Mechanica Sinica/Lixue Xuebao **37**(12), 1718–1726 (2021) https://doi.org/10.1007/s10409-021-01143-6

[15] Im, S., Lee, J., Cho, M.: Surrogate modeling of elasto-plastic problems via long short-term memory neural networks and proper orthogonal decomposition. Computer Methods in Applied Mechanics and Engineering **385** (2021) https://doi.org/10.1016/j.cma.2021.114030

[16] Borst, R., Crisfield, M., Remmers, J., Verhoosel, C.: Non-Linear Finite Element Analysis of Solids and Structures: Second Edition, 2nd edn. Wiley, Chichester (2012). https://doi.org/10.1002/9781118375938

[17] Mohamadnejad, S., Basti, A., Ansari, R.: Analyses of Dislocation Effects on Plastic Deformation. Multiscale Science and Engineering **2**(2-3), 69–89 (2020) https://doi.org/10.1007/s42493-020-00037-2

[18] Narkhede, M.V., Bartakke, P.P., Sutaone, M.S.: A review on weight initialization strategies for neural networks. Artificial Intelligence Review **55**(1), 291–322 (2022) https://doi.org/10.1007/s10462-021-10033-z

[19] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (2014) https://doi.org/10.48550/arXiv.1412.6980

[20] Ribeiro, A.H., Tiels, K., Aguirre, L.A., Schön, T.B.: Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness (2019) https://doi.org/10.48550/arXiv.1906.08482

[21] Hochreiter, S., Schmidhuber, J.: Long Short Term Memory. Neural Computation **9**(8), 1735–1780 (1997) https://doi.org/10.1162/neco.1997.9.8.1735

[22] Crawshaw, M.: Multi-Task Learning with Deep Neural Networks: A Survey (2020) https://doi.org/10.48550/arXiv.2009.09796

[23] Caruana, R., Pratt, L., Thrun, S.: Multitask Learning. Technical report (1997). https://doi.org/10.1023/A:1007379606734

[24] Baxter, J.: A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling. Machine Learning **28**(1), 7–39 (1997) https://doi.org/10.1023/A:1007327622663

[25] Yu, T., Zhu, H.: Hyper-Parameter Optimization: A Review of Algorithms and Applications (2020) https://doi.org/10.48550/arXiv.2003.05689

[26] Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A Next-generation Hyperparameter Optimization Framework (2019) https://doi.org/10.48550/arXiv.1907.10902

[27] Watanabe, S.: Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance (2023) https://doi.org/10.48550/arXiv.2304.11127