applied sciences

MDPI

Article

Mesh-Free Surrogate Models for Structural Mechanic FEM Simulation: A Comparative Study of Approaches

Johannes G. Hoffer [1,*], Bernhard C. Geiger [1], Patrick Ofner [1] and Roman Kern [2,*]

[1] Know-Center GmbH, Research Center for Data-Driven Business & Big Data Analytics, Inffeldgasse 13, 8010 Graz, Austria; bgeiger@know-center.at (B.C.G.); patrick@ofner.science (P.O.)

[2] Institute of Interactive Systems and Data Science, Graz University of Technology, Inffeldgasse 16c, 8010 Graz, Austria

* Correspondence: jhoffer@know-center.at (J.G.H.); rkern@tugraz.at (R.K.)

Abstract: The technical world of today fundamentally relies on structural analysis in the form of design and structural mechanic simulations. A traditional and robust simulation method is the physics-based finite element method (FEM) simulation. FEM simulations in structural mechanics are known to be very accurate; however, the higher the desired resolution, the more computational effort is required. Surrogate modeling provides a robust approach to address this drawback. Nonetheless, finding the right surrogate model and its hyperparameters for a specific use case is not a straightforward process. In this paper, we discuss and compare several classes of mesh-free surrogate models based on traditional and thriving machine learning (ML) and deep learning (DL) methods. We show that relatively simple algorithms (such as k-nearest neighbor regression) can be competitive in applications with low geometrical complexity and extrapolation requirements. With respect to tasks exhibiting higher geometric complexity, our results show that recent DL methods at the forefront of literature (such as physics-informed neural networks) are complicated to train and to parameterize and thus, require further research before they can be put to practical use. In contrast, we show that already well-researched DL methods, such as the multi-layer perceptron, are superior with respect to interpolation use cases and can be easily trained with available tools. With our work, we thus present a basis for the selection and practical implementation of surrogate models.

Keywords: FEM; surrogate modeling; mesh-free; machine learning; deep learning

Citation: Hoffer, J.G.; Geiger, B.C.; Ofner, P.; Kern, R. Mesh-Free Surrogate Models for Structural Mechanic FEM Simulation: A Comparative Study of Approaches. *Appl. Sci.* **2021**, *11*, 9411. https://doi.org/10.3390/app11209411

Academic Editors: Jin-Gyun Kim, Jae-Hyuk Lim and Peter Persson

Received: 15 September 2021
Accepted: 3 October 2021
Published: 11 October 2021

1. Introduction

Assessing the properties of mechanical structures with real physical experiments is expensive, as it costs both time and resources. To reduce these costs of knowledge enrichment in the field of structural analysis, computer simulations of structural mechanics have become crucial. An essential simulation method is the finite element method (FEM) in which the simulation domain space is represented by a finite number of connected elements. Space- and time-dependent behavior between connected elements and within the elements themselves is governed by physical equations. Observation of real physical experiments provides the coefficients for these governing equations. Since most geometries and use cases cannot be solved analytically, an approximation of the proposed physical equations is obtained by numerical methods [1]. However, solving complex problems with FEM is time-consuming and computationally expensive. In order to reduce the computational effort, surrogate modeling offers a promising solution [2].

Surrogate models are trained in a supervised manner and are designed to learn the function mapping between inputs and outputs from a given FEM simulation use case. With a sufficient amount of training data with respect to the use case, an according model is able to substitute for the FEM simulation use case up to a certain accuracy.

There is already a considerable number of related work concerning surrogate modeling of structural mechanics simulations with machine learning (ML) or deep learning (DL) approaches. In the following, we want to present the most important works for this paper. Artificial neural networks (ANN) are used in the work of Roberts et al. [3] to predict damage development in forged brake discs reinforced with Al-SiC particles, using damage maps. The ANN is a multilayer perceptron (MLP), and training data are obtained from FEM simulations using the commercial DEFORM simulation software. For rapid estimation of forming and cutting forces for given process parameters, Hans Raj et al. [4] investigate a method using MLP models. The researchers focus on two processes: hot upsetting and extrusion. Each process, represented by a MLP, is trained with FEM simulation results from the FORGE2 commercial FEM simulation software. García-Crespo et al. [5] predict the projectile response after impact with steel armor using a MLP; their surrogate model studied is trained with data from FEM simulations of the use case. Nourbakhsh et al. [6] explore generalizable surrogate models for 3D trusses, using MLP and FEM training data. Chan et al. [7] estimate the performance of hot-forged product designs, using a MLP trained on FEM results obtained with the commercial software DEFORM. D'Addona and Antonelli [8] use single-layer feedforward ANNs instead of FEM as a metamodel in a sequential approximate optimization (SAO) algorithm. In a case study on hot forging of a steel disk, they compare their results with an ANN trained on FEM simulation results and the FEM simulation software QForm3D. Gudur and Dixit [9] predict the velocity field and location of neutral point of cold flat rolling with a MLP trained with rigid-plastic FEM simulation results. Pellicer-Valero et al. [10] predict the mechanical behavior of different livers with MLPs trained from FEM simulations.

Abueidda et al. [11] estimate the mechanical properties of a two-dimensional checkerboard composite using a convolutional neural network (CNN) trained with FEM results. Regarding mesh-based approaches, Pfaff et al. [12] present a framework to train graph neural networks (GNN) on mesh-based simulations and show the applicability in aerodynamics, structural mechanics, and fabric.

Surrogate models were also obtained using classical, i.e., non-neural ML, approaches. For example, the authors of [3] apply Gaussian process regression (GPR) besides ANN in their approach. Loghin and Ismonov [13] predict the stress intensity factors, using GPR trained with FEM results of a classical bolt-nut assembly. Ming et al. [14] model the electrical discharge machining process with GPR trained from data generated with numerical FEM simulation.

Using support vector regression (SVR), Pan et al. [15] construct a metamodel in an optimization approach for lightweight vehicle design. Training data are generated, using design of experiment approaches with FEM simulations. To predict the stress at the implant–bone interface, Li et al. [16] utilize SVR in order to replace FEM simulation. Hu and Li [17] estimate cutting coefficients in a mechanistic milling force model with SVR trained with FEM simulation data.

Employing tree-based models, Martínez-Martínez et al. [18] estimate the biomechanical behavior of breast tissue under compression, using three different tree-based models trained from FEM simulations. The models are trained with FEM data in terms of nodal coordinates and nodal tissue membership. Zhang et al. [19] estimate the base failure stability for braced excavations in anisotropic clay using extreme gradient boosting, random forest regression (RFR) and data obtained from FEM simulation results. Qi et al. [20] utilize a decision tree regressor to predict the mechanical properties of carbon fiber reinforced plastics with data obtained from FEM simulations. Besides MLPs Pellicer-Valero et al. [10] utilize RFRs to predict the biomechanics of livers.

A recent neural network–based approach are physics informed neural networks (PINNs). PINNs are trained simultaneously on data and governing differential equations and can be used for the solution and inversion of equations governing physical systems. Utilizing PINNs, Haghighat and Juanes [21] substitute a particular FEM simulation of a perforated strip under uniaxial extension. In [22], Haghighat et al. present a surrogate modeling approach with PINNs and a specific use case. Focusing on consistency, Shin [23] evaluates findings regarding PINNs with Poisson's equation and the heat equation. Yin et al. [24] use PINNs to predict permeability and viscoelastic modulus from thrombus deformation data, described by the fourth-order Cahn–Hilliard and Navier–Stokes equations. In addition to the application of PINNs in structural mechanics problems, there is also a considerable number of papers, especially in computational fluid dynamics [25–29].

Related work shows capabilities of surrogate modeling, thus demonstrating the feasibility of supervised learning models trained with FEM simulations. From our analysis of the existing literature, we identify the following drawbacks:

- In most cases, the surrogate model only substitutes for a subset of the considered computational domain. Thus, such an approach focuses only on a region of interest and cannot be used to evaluate the entire computational domain (notable exceptions are [12,22]).
- Surrogate models representing the complete discretized computational domain (mesh) are solely fitted and evaluated on one use case—generalization to unseen data is only achieved with respect to the discretization of the computational domain, but not with respect to other use case specific parameters (notable exception concerning material parameters [22]).
- Due to differences in FEM use cases and data, the comparison of related work is useful only in some cases.
- Replication of published experiments is often not achievable because important parameters are not reported, e.g., number of finite elements, type of finite elements (bilinear, biquadratic, reduced integration etc.), method of discretization (meshing), as well as hyperparameters of the ML models, such as learning or activation functions.

To address these drawbacks, we present the following contributions of our paper:

1. We present the main DL and ML methods together with a compact description and mathematical notation to equip practitioners with a reference to surrogate FEM simulation mesh-free and assess the feasibility and maturity of the novel PINNs method.
2. We utilize three classic use cases in structural mechanics and evaluate these models in terms of performance on unseen configurations (inter- and extrapolation) in order to assess their ability to generalize across different use case specific parameters.
3. We discuss the characteristics of all DL and ML models, and their practical implications, in the context of the use cases.

With our work, we pave the way of mesh-free surrogate modeling for practical use: we provide a basis for efficient model and hyperparameters selection regarding use case and performance metrics. These insights shall not only assist the domain expert during model selection, but will also help in consolidating the current research in mesh-free surrogate modeling for structural mechanics applications.

We report all information to make our experiments reproducible. If certain model settings are not mentioned, they are left at default values. Moreover, our FEM simulations are performed with Abaqus Student Edition 2019 (Dassault Systèmes, Velizy-Villacoublay, France), and thus, the process of data generation is not limited to commercial software, which makes it possible for everyone to connect to our research.

The remainder of this paper is organized as follows. In Section 2, we present the materials and methods of our experiments, first providing insights into the process of data generation, using the FEM simulations in Section 2.1, then describing the datasets obtained from the FEM simulations in Section 2.2, followed by the ML and DL models used in Section 2.3. Section 3 shows the results, which are discussed in Section 4. In Section 5, we present the conclusion of our work and an outlook for the future.

2. Materials and Methods

In this section, we present all relevant information about the methodology of our experiments. First, Section 2.1 provides an overview of the data generation process, using three classic FEM simulation use cases. Then, Section 2.2 describes the datasets used from the FEM simulations, and Section 2.3 presents the ML and DL models used. A more detailed overview of the mathematical background and assumptions of the ML and DL models can be found in the Appendix. When predicting a particular use case with a surrogate model, the individual nodes discretizing the particular geometry of the use case (i.e., mesh) are sequentially input into the surrogate model with the appropriate generalization variable. The surrogate model then predicts the output of each node in sequence; see Figure 1.

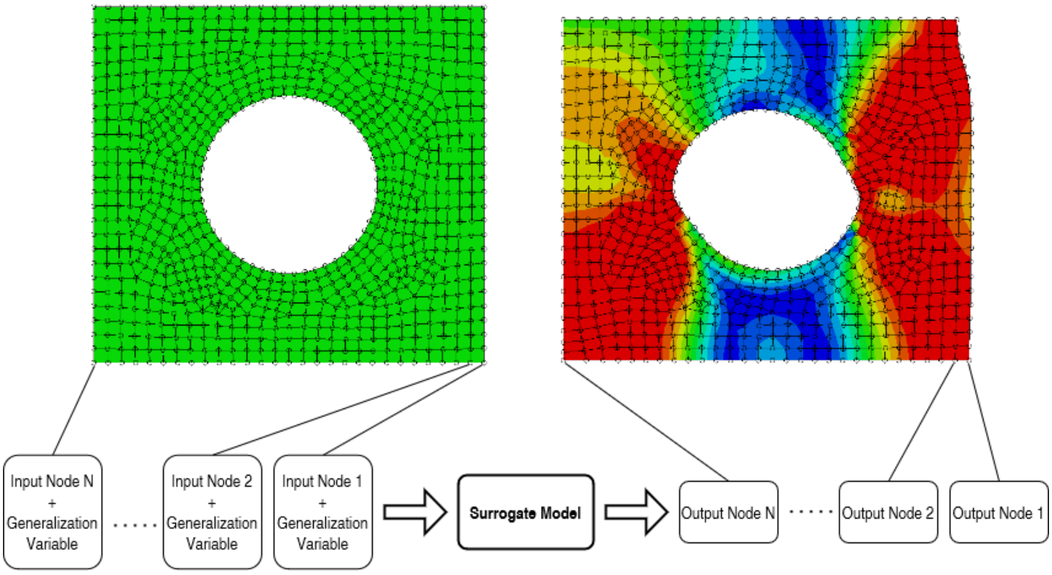


Figure 1. Principle of our surrogate model approach: all N nodes (i.e., their coordinates), together with the respective generalization variable, are sequentially entered into a surrogate model, which then sequentially predicts the outcome of the respective coordinates (i.e., the displacements, strains, and stresses of the respective node).

It should be noted that there are no constraints on the discretization (mesh), i.e., the node coordinates can be freely chosen within the simulation domain and nodes are not connected to each other. Therefore, we refer to our approach as mesh-free, but we want to clearly distinguish ourselves from other mesh-free methods, such as smoothed particle hydrodynamics, the diffuse element method, the moving particle finite element method, etc. The predictions of the individual nodes together constitute the prediction for the simulation domain of the particular use case. By adding the nodal displacement outputs of the surrogate model to the initial node coordinates, we obtain the new deformed geometry. Further surrogate model outputs (e.g., stresses, strains) describe the queried nodes and thus the complete simulation domain in more detail.

2.1. FEM Use Cases

For illustration, we base our evaluation on three classic use cases from structural mechanics. We consider the (1) tensile load, (2) bending load and (3) compressive load:

1. Elongation of a plate with a perforation;
2. Bending of a beam;
3. Compression of a block with four perforations.

See Table 1 and Figure 2. We utilize an isotropic elasto-plastic rate-independent material model (i.e., a perfectly plastic material). The kinematic relations for our 2D plane strain use cases are defined by the total strain components $\varepsilon_{xx} = \frac{\partial u_x}{\partial x}$, $\varepsilon_{yy} = \frac{\partial u_y}{\partial y}$, $\varepsilon_{xy} = \frac{1}{2}(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x})$, $\varepsilon_{zz} = 0$ with displacements u_x and u_y and deviatoric strain components $e_{xx} = \varepsilon_{xx} - \frac{\varepsilon_{vol}}{3}$, $e_{yy} = \varepsilon_{yy} - \frac{\varepsilon_{vol}}{3}$, $e_{xy} = \varepsilon_{xy}$ and $e_{zz} = -\frac{\varepsilon_{vol}}{3}$. Since there is no volumetric plastic strain in the von Mises yield function, the volumetric strain can be expressed as

$\varepsilon_{vol} = \text{trace}(\boldsymbol{\varepsilon})$ s.t. $\varepsilon_{vol} = \varepsilon_{xx} + \varepsilon_{yy}$. The deviatoric stress components are defined by $s_{xx} = \sigma_{xx} - (\frac{\sigma_{xx}+\sigma_{yy}+\sigma_{zz}}{3})$, $s_{yy} = \sigma_{yy} - (\frac{\sigma_{xx}+\sigma_{yy}+\sigma_{zz}}{3})$, $s_{xy} = \sigma_{xy}$ and $s_{zz} = \sigma_{zz} - (\frac{\sigma_{xx}+\sigma_{yy}+\sigma_{zz}}{3})$, where σ_{ij} $(i, j \in \{x, y\})$ are the components of the Cauchy stress tensor. The plastic strain components are defined by $\varepsilon^{pl}_{xx} = \bar{\varepsilon}^{pl} \frac{3}{2} \frac{s_{xx}}{q}$, $\varepsilon^{pl}_{yy} = \bar{\varepsilon}^{pl} \frac{3}{2} \frac{s_{yy}}{q}$, $\varepsilon^{pl}_{xy} = \bar{\varepsilon}^{pl} \frac{3}{2} \frac{s_{xy}}{q}$ and $\varepsilon^{pl}_{zz} = \bar{\varepsilon}^{pl} \frac{3}{2} \frac{s_{zz}}{q}$ with equivalent plastic strain of the von Mises model as $\bar{\varepsilon}^{pl} = \bar{\varepsilon} - \frac{\sigma_Y}{3\mu} \geq 0$, where σ_Y is the yield stress and μ the second Lamé parameter. The total equivalent strain is defined by $\bar{\varepsilon} = \sqrt{\frac{2}{3} \sum_{i,j \in \{x,y\}} e_{ij} e_{ij}}$ with deviatoric strain components e_{ij}. The decomposition of the strain is $\varepsilon_{ij} = \varepsilon^{el}_{ij} + \varepsilon^{pl}_{ij}$ with elastic component ε^{el}_{ij} and plastic component ε^{pl}_{ij} of the respective strain matrices. The equivalent stress is defined by $q = \sqrt{\frac{3}{2} s_{ij} s_{ij}}$. In our PINN approach, we utilize the definitions of the total strain components, deviatoric strain and stress components and plastic strain components in the respective regularization term.

We use quarter symmetry in use cases 1 and 3 to make efficient use of computational resources. Additional information regarding the variation of parameters in the simulations is presented in Table 2, where simulations marked in bold are used for the test and evaluation of the surrogate models and are not in the training dataset. Conversely, simulations not marked in bold represent the training dataset and are not in the test dataset. In use cases exhibiting varied geometry parameters (i.e., elongation of a plate and compression of a block use cases), the mesh is also different in each simulation. Thus, we train and evaluate the surrogate models on use cases with different meshes (i.e., in each simulation, the node coordinates differ).

Table 1. Classic FEM use cases. Overview of the three use cases and their main change and types of deformations. In the first two use cases, only a single change is conduced, while in the last use case, a combination of changes is studied.

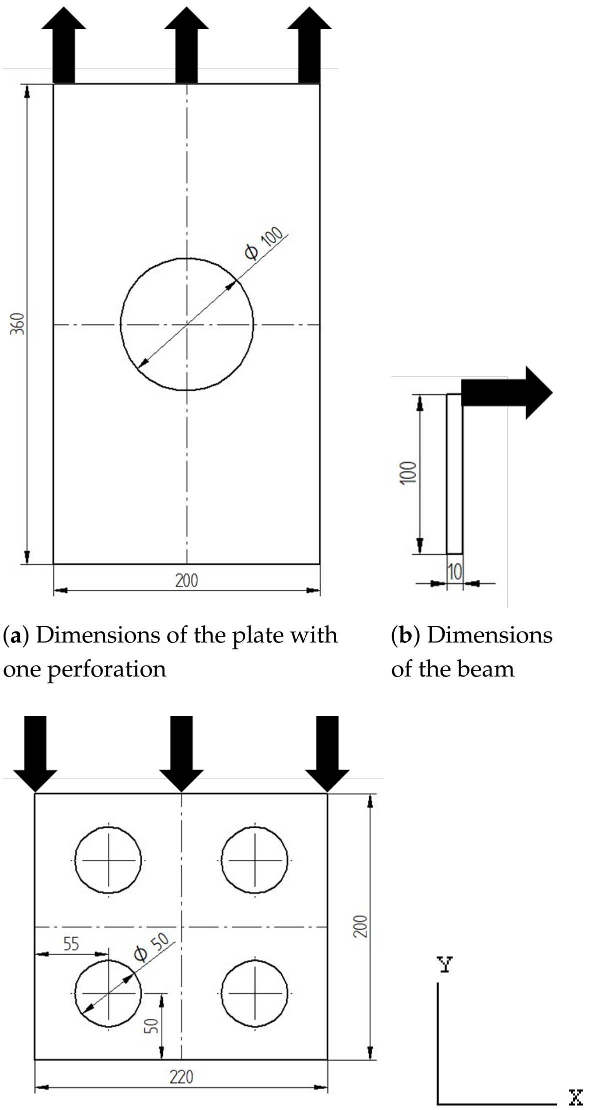
Use Case	Change	Deformation
Plate	Geometry	Elongation
Beam	Material Properties	Bending
Block	Geometry, Material Properties	Compression

The first use case, a perforated steel strip under tensile load, is similar to the nonlinear solid mechanics use case of [21,22]. However, in our approach, we evaluate the generalization over the perforation diameter and use material properties for steel and a top edge displacement of 5 mm in positive y-axis to consider a more challenging use case.

We execute different simulation settings, where the generalization variable (diameter of perforation) is changed in each simulation; see Figure 2a and Table 2. In our second use case, we simulate a bending beam that end is displaced about 5 mm in the positive x-direction; see Figure 2b. We vary the yield stress generalization variable in each simulation setting; see Table 2. In our third use case, we simulate a quarter-symmetric block with four perforations under compressive load, which is compressed about 5 mm in the negative y-axis; see Figure 2c. In this use case, we vary two generalization variables (yield stress and width of the block) in each simulation; see Table 2.

We evaluate our models on interpolation (i.e., that the generalization variables for testing are within the range of the generalization variables observed during training) and extrapolation (i.e., that the generalization variables for testing are outside the range of the generalization variables observed during training) tasks. In Table 2, we mark interpolation tasks with superscript (i) and extrapolation tasks with superscript (e).

In Figure 3, we present the perfect nonlinear elastoplastic material behavior of our use cases. The Young's modulus is 210 GPa, Poisson's ratio 0.3 and the yield stress 900 MPa. In our first use case, the perforated plate, we use this setting in each simulation. In the other two use cases, the yield stress varies, while the remaining material parameters stay the same.



(**a**) Dimensions of the plate with one perforation

(**b**) Dimensions of the beam

(**c**) Dimensions of the block with four perforations

(**d**) Coordinate system

Figure 2. The three use cases: (**a**) elongation of a plate (diameter = 100 mm) about 5 mm at the top end in positive y-direction, (**b**) bending of a beam by a displacement at the top end about 5 mm in positive x-direction, (**c**) compression of a block with four perforations in the center of the quarter-symmetric parts (width = 220 mm) about 5 mm in negative y-direction and (**d**) the considered coordinate system.

All parts are meshed, using plane strain 4-node bilinear quadrilateral elements with reduced integration and hourglass control. Please note that although [22] recommends the use of larger order elements for the approximation of body forces, we use bilinear elements since we do not use body forces in our surrogate modeling approaches. We create a finer mesh near additional geometric details (i.e., perforations in the plate and block use cases) and seed the perforation edge of the plate with an approximate size of 3.8 mm and the remaining edges with an approximate size of 5 mm. The perforation edges of the block are seeded with an approximate size of 3 mm and the remaining edges with an approximate size of 4 mm. The beam exhibits no comparable geometric details; thus, we seed all edges with an approximate size of 1.5 mm.

Table 2. Dataset generation by executing several different simulations with varying generalization variables (Plate: perforation *Diameter*, Beam: *Yield Stress* and Block: *Yield Stress* and *Width*), bold marked simulations are not in the training dataset and only used for test and evaluation. Interpolation tasks are marked with superscript (i) and extrapolation tasks with superscript (e).

Plate													
Simulation ID	**1** $^{(e)}$	2	3	**4** $^{(i)}$	5	**6** $^{(i)}$	7	8	**9** $^{(e)}$				
Diameter [mm]	**60**	70	80	**90**	100	**110**	120	130	**140**				
Beam													
Simulation ID	**1** $^{(e)}$	2	3	**4** $^{(i)}$	5	**6** $^{(i)}$	7	8	**9** $^{(e)}$				
Yield Stress [MPa]	**850**	900	950	**1000**	1050	**1100**	1150	1200	**1250**				
Block													
Simulation ID	**1** $^{(e)}$	**2** $^{(e)}$	3	4	5	6	**7** $^{(i)}$	8	9	10	11	**12** $^{(e)}$	**13** $^{(e)}$
Yield Stress [MPa]	**750**	**750**	900	900	900	1050	**1050**	1050	1200	1200	1200	**1350**	**1350**
Width [mm]	**180**	**260**	200	220	240	200	**220**	240	200	220	240	**180**	**260**

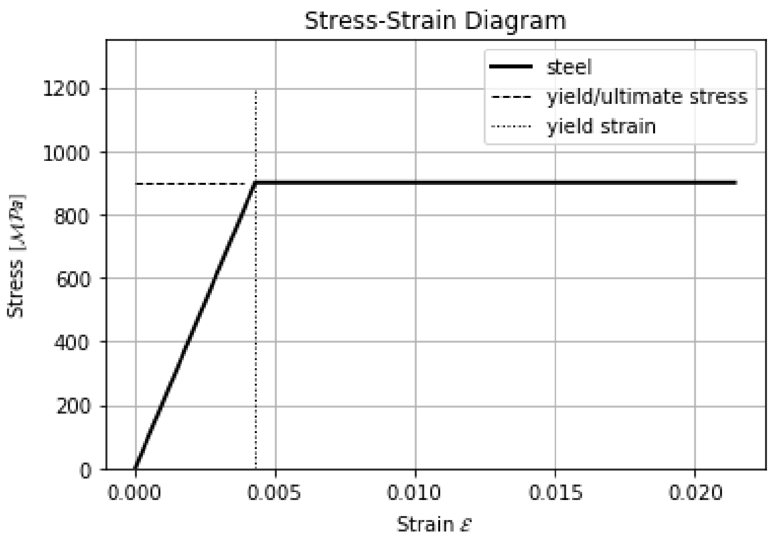


Figure 3. Perfect nonlinear elastoplastic material properties for a Young's modulus of 210 GPa, Poisson's ratio of 0.3 and yield stress of 900 MPa. The yield stress varies in simulations regarding the beam and block use cases.

We obtain our FEM simulation results in the context of general static simulations. Details of the simulation steps are shown in Table 3. Simulation control parameters that are not listed are left at default values.

Table 3. Abaqus FEM simulation control parameters.

Abaqus FEM Simulation Settings	
Simulation type	Static, General
Time period	1
Nlgeom	On
Max number of increments	100
Initial increment size	1
Min increment size	1×10^{-5}
Max increment size	1
Equation solver method	Direct
Solution technique	Full Newton

2.2. Dataset

The nodal data from our Abaqus FEM simulations constitute the datasets. For each use case, the nodal data are split into training and test dataset, respectively. The training dataset $D = \{X_1, \ldots, X_n\}$ with number of training instances n and the test dataset $T = \{X_{n+1}, \ldots, X_{n+m}\}$ with number of test instances m are generated from several FEM simulations; see Tables 2 and 6, where bold marked simulations belong to T and the remaining to D. Thus, we split our data due to different generalization variables and not randomly. We denote each instance with index i, $i \in \{1, 2, \ldots, n+m\}$. An instance $X_i = (x_i, y_i)$ is generated of an input vector $x_i \in \mathbb{R}^p$ and output vector $y_i \in \mathbb{R}^q$. Each input vector x_i is composed of the initial x- and y-coordinates of a FEM node and the respective generalization variable (i.e., perforated plate: *Diameter*, beam: *Yield Stress*, block with four perforations: *Width* and *Yield Stress*) of the FEM simulation; see Table 4. Thus, we have $p = 3$ in the plate and beam use case, and $p = 4$ in the block use case.

Table 4. Surrogate model input variables. Data obtained from FEM simulations are transformed so that each FEM node (represented by its x- and y-coordinates) with the respective generalization variable is an instance.

Simulation	Plate	Beam	Block
Input variables	x-coordinate y-coordinate Diameter	x-coordinate y-coordinate Yield Stress	x-coordinate y-coordinate Yield Stress Width

In our setting, each output vector y_i contains 13 ($q = 13$) output variables obtained from FEM simulation with input x_i, namely the ε^t_{xx}, ε^t_{xy} and ε^t_{yy} total strain components, the ε^p_{xx}, ε^p_{xy}, ε^p_{yy} and ε^p_{zz} plastic strain components, the σ_{xx}, σ_{xy}, σ_{yy} and σ_{zz} principal and shear stress components and the displacement in x- and y-directions u and v of each node; see Table 5 and Figure 4. We split the data in a training and test dataset (see Table 6) and standardized the data by removing the mean and scaling to unit variance.

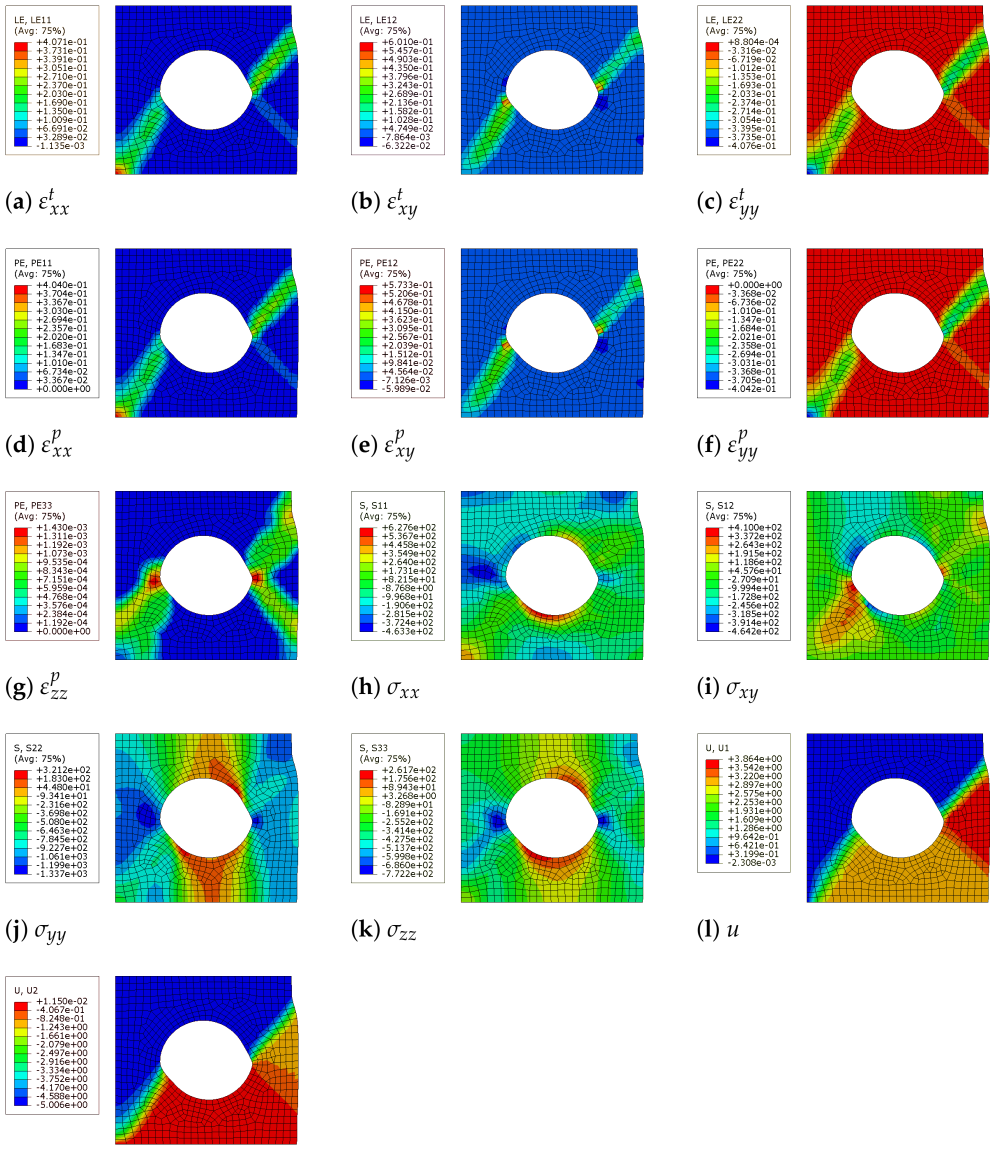
Table 5. Surrogate model output variables. For each input FEM node, a surrogate model predicts its respective strains, stresses and displacements.

Output Variables			
ε^t_{xx}	ε^p_{xx}	σ_{xx}	u
ε^t_{xy}	ε^p_{xy}	σ_{xy}	v
ε^t_{yy}	ε^p_{yy}	σ_{yy}	
	ε^p_{zz}	σ_{zz}	

In Figure 4, we present graphical results with visible mesh obtained from Abaqus FEM simulation of the output variables used for a block use case.

Table 6. Dataset splits: number of training instances n and test instances m due to the data generation from Table 2.

	Plate	Beam	Block
Training dataset D	4447	2720	6722
Test dataset T	3534	2176	4107



(a) ε^t_{xx} **(b)** ε^t_{xy} **(c)** ε^t_{yy}

(d) ε^p_{xx} **(e)** ε^p_{xy} **(f)** ε^p_{yy}

(g) ε^p_{zz} **(h)** σ_{xx} **(i)** σ_{xy}

(j) σ_{yy} **(k)** σ_{zz} **(l)** u

(m) v

Figure 4. Block use case: Abaqus FEM results that our surrogate models should predict.

2.3. Surrogate Models

In this section, we give an overview of the surrogate models used and their general assumptions; to highlight the differences as well as the advantages and disadvantages between them, we present a detailed mathematical background in Appendix A. We have selected models from different learning paradigms:

1. Gradient boosting decision tree regressor (GBDTR): piecewise constant model.
2. K-nearest neighbor regressor (KNNR): distance-based model.
3. Gaussian process regressor (GPR): Bayesian model.
4. Support vector regressor (SVR): hyperplane-based model.
5. Multi layer perceptron (MLP): classic feedforward neural network model.
6. Physics informed neural network (PINN): neural network model with physics-based regularization.

3. Results

For evaluation, we split the data into a training and test dataset to fit and test our surrogate models; see Table 6 for the dataset sizes and Table 2 for more details regarding the data split.

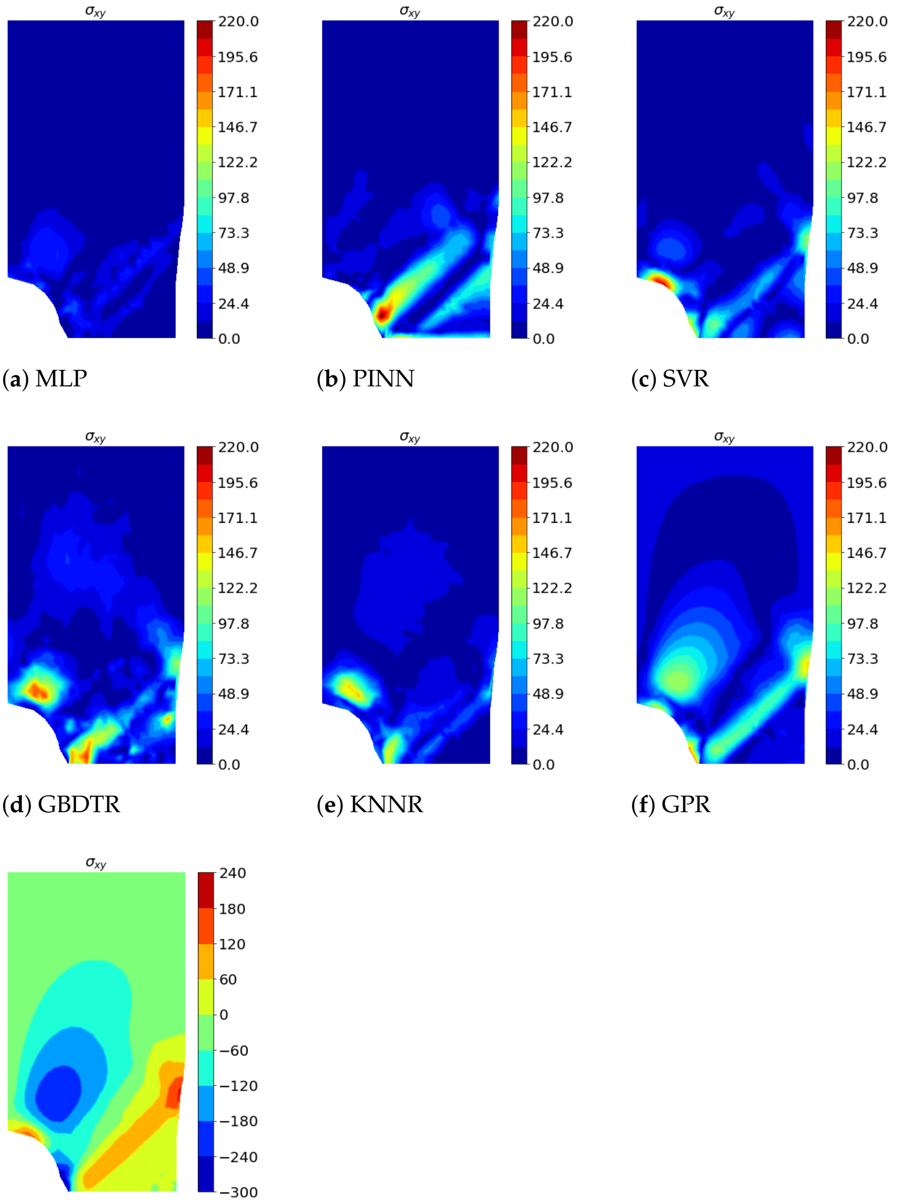
As a next step, we need to define hyperparameters for each model and each use case. We performed hyperparameter optimization using only training data; no test data were used. In our PINN approaches, the adaptation of hyperparameters was based on the work of [21,22]. Our MLPs were designed to be similar to our PINNs to allow for fair comparisons. We varied hyperparameters in our neural network approaches (MLP and PINN) following best practices and guidelines, where we optimized the number of hidden layers, number of neurons per hidden layer, activation function, validation split, earlystopping patience and the size of the batch per training epoch. Regarding the rest of our models, we applied a grid-search with a five fold cross-validation, utilizing the training data to obtain the best hyperparameters. The hyperparameters for each use case are in Appendix B and Tables A1–A6.

Our evaluation is based on R2-scores with respect to the FEM results and inference time. For models that contain inherent randomness, such as MLPs, GBDTR and PINNs, a five-fold cross-validation was conducted. For these models, we report the mean values and standard deviation of the R2-score. For the sake of brevity, we report only the average R2-scores across all 13 targets in this section; see Tables 7–9. The R2-scores for individual targets are provided in Appendix C. The inference times are based on the mean value of three measurements. Inferences were run on a machine with 16 GB RAM, 8 CPUs and Intel(R) i7-8565 2.0GHz processor. To compare the inference time of our surrogate models with the computation time required to run FEM simulations, we have included the latter also in Tables 7–9.

Table 7. Plate: averaged results, bold values indicate the best performing surrogate models. Values in parentheses are the corresponding standard deviations of the average R2-scores due to repeated experiments of stochastic process models. For further information concerning simulations, see Table 2.

Model	MLP	PINN	SVR	GBDTR	KNNR	GPR	FEM
	Simulation 1						
R2	**0.9900** (6.155×10^{-9})	0.7797 (8.709×10^{-2})	0.6188	0.6606 (3.959×10^{-8})	0.8164	0.6131	-
Inference time [s]	0.0523	0.0746	1.15	0.311	0.00722	0.151	9.01
	Simulation 4						
R2	**0.9978** (1.970×10^{-4})	0.9089 (2.598×10^{-2})	0.7174	0.9014 (6.310×10^{-2})	0.9298	0.8761	-
Inference time [s]	0.0781	0.0638	1.20	0.271	0.00734	0.139	9.08
	Simulation 6						
R2	**0.9920** (1.889×10^{-3})	0.8470 (6.309×10^{-2})	0.7251	0.8503 (1.005×10^{-1})	0.9219	0.8676	-
Inference time [s]	0.0595	0.0641	1.10	0.251	0.00797	0.131	9.88
	Simulation 9						
R2	**0.9786** (2.970×10^{-5})	0.7562 (1.046×10^{-1})	0.6568	0.7263 (9.780×10^{-9})	0.8045	0.5651	-
Inference time [s]	0.0665	0.0715	1.02	0.251	0.00734	0.139	10.03

For graphical results, we chose simulations that cover the error situation quite well in order to make statements about the performance of each model. In addition to the absolute errors (Figures 5a–f–10a–f), the corresponding FEM simulations of the basis are shown in Figures 5g–10g.



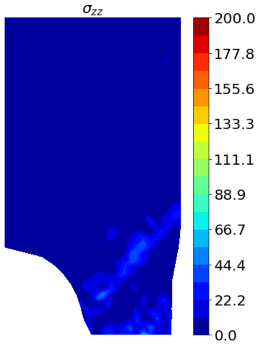
(**a**) MLP (**b**) PINN (**c**) SVR

(**d**) GBDTR (**e**) KNNR (**f**) GPR

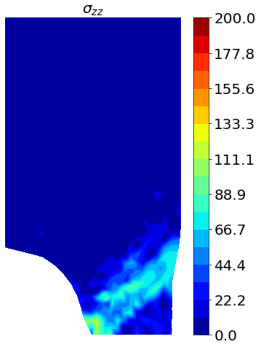
(**g**) FEM result

Figure 5. Elongation of a perforated plate, Simulation 1 (extrapolation): absolute errors of different surrogate models (**a**–**f**) and ground truth Abaqus FEM simulation (**g**) of σ_{xy}.

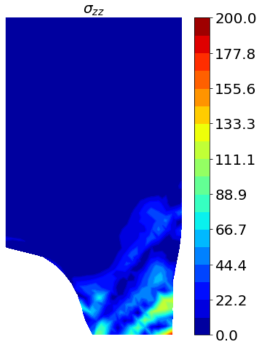
GBDTR, KNNR, GPR and SVR algorithms were implemented with the scikit-learn library version 0.24.0 in Python. The SVR and GBDTR algorithms were constructed with MultiOutputRegressor scikit-learn API to fit one regressor per target. Regarding our DL algorithms, the utilized MLPs were implemented with the keras API version 2.4.3 and our PINNs were implemented with the sciann API version 0.5.5.0 in Python 3.8.5. We used the PDEs from [21,22], but instead of the inversion part, we trained our PINNs additionally with plastic strain data, same as for the rest of the surrogate models.



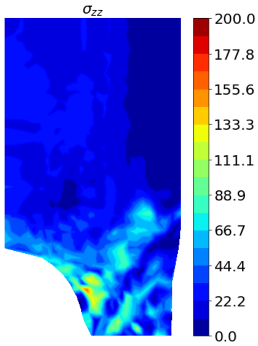
(**a**) MLP



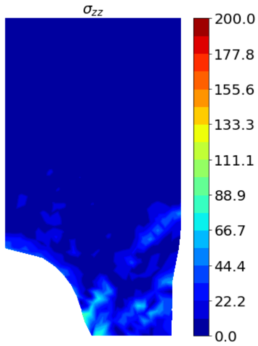
(**b**) PINN



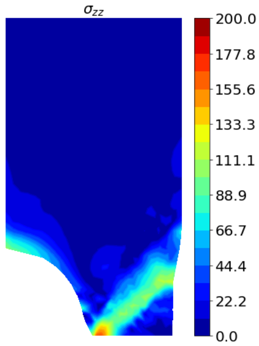
(**c**) SVR



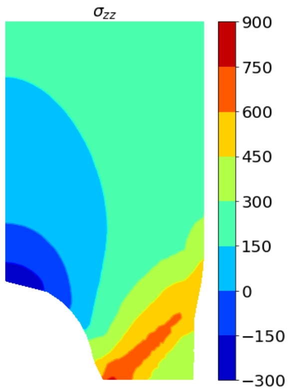
(**d**) GBDTR



(**e**) KNNR



(**f**) GPR



(**g**) FEM result

Figure 6. Elongation of a perforated plate, Simulation 4 (interpolation): absolute errors of different surrogate models (**a**–**f**) and ground truth Abaqus FEM simulation (**g**) of σ_{zz}.

In the elongation of a perforated plate use case, our approach is based on a total of nine FEM simulations. We used five simulations for training and four simulations to evaluate the fitted models; see Table 2. We report the average of R2-scores across all outputs in Table 7 with the corresponding inference times.

Regarding extrapolation, the absolute errors of each surrogate model with respect to σ_{xy} of Simulation 1 are shown in Figure 5. We plot the absolute errors of each surrogate model of σ_{zz} of Simulation 4 in Figure 6 as an example of interpolation. In addition, we show in both figures the ground truth of the corresponding output variable obtained from the FEM simulation. For both interpolation and extrapolation, the errors are large near the shear band. As far as extrapolation is concerned, in addition to the errors near the shear band, most models have significant errors near the maximum negative xy shear stresses; see blue areas in Figure 5g. GBDTR performs well overall, though the error increases in various locations; while PINNs have a similar average performance, they perform better outside the shear band regarding absolute errors. MLP overall shows the best results followed by KNNR.

In the bending beam use case, similar to the perforated plate use case, we trained our models on five simulations and tested them using the remaining four, see Table 2. We present the average R2-scores across all outputs and inference times in Table 8 for the test simulations 1, 4, 6 and 9.

Table 8. Beam: averaged results, bold values indicate the best performing surrogate models. Values in parentheses are the corresponding standard deviations of the average R2-scores due to repeated experiments of stochastic process models. For further information concerning simulations, see Table 2.

Model	MLP	PINN	SVR	GBDTR	KNNR	GPR	FEM
	Simulation 1						
R2	0.6682 (7.345×10^{-6})	0.6165 (1.648×10^{-2})	0.5122	**0.7120** (1.088×10^{-8})	0.6288	0.5377	-
Inference time [s]	0.0781	0.0638	1.20	0.271	0.00734	0.139	9.08
	Simulation 4						
R2	**0.9979** (1.319×10^{-3})	0.9379 (1.042×10^{-3})	0.7558	0.9640 (6.751×10^{-4})	0.9621	0.8243	-
Inference time [s]	0.0457	0.110	0.418	0.0541	0.0790	0.221	6.81
	Simulation 6						
R2	**0.9981** (8.315×10^{-4})	0.9314 (8.396×10^{-4})	0.7406	0.9516 (1.368×10^{-4})	0.9617	0.8059	-
Inference time [s]	0.0442	0.0668	0.402	0.0569	0.0705	0.212	6.61
	Simulation 9						
R2	−1196.2920 (6.166×10^{3})	−1305.9226 (3.269×10^{1})	**−107.7646**	−830.8926 (4.2984)	−322.4940	−420.9029	-
Inference time [s]	0.0781	0.0638	1.20	0.271	0.00734	0.139	9.08

We provide a graphical representation of the absolute error of the surrogate models regarding ε^{t}_{yy} in Figure 7a–f with the FEM simulation result in (g) as one instance of interpolation. Absolute errors of the surrogate models regarding ε^{p}_{xx} and extrapolation are shown in Figure 8. Overall higher errors can be observed near the encastred boundary condition of the beam for some models for that output. While the PINN shows a competitive average R2-score regarding interpolation, on this single target, its performance shows significant weaknesses.

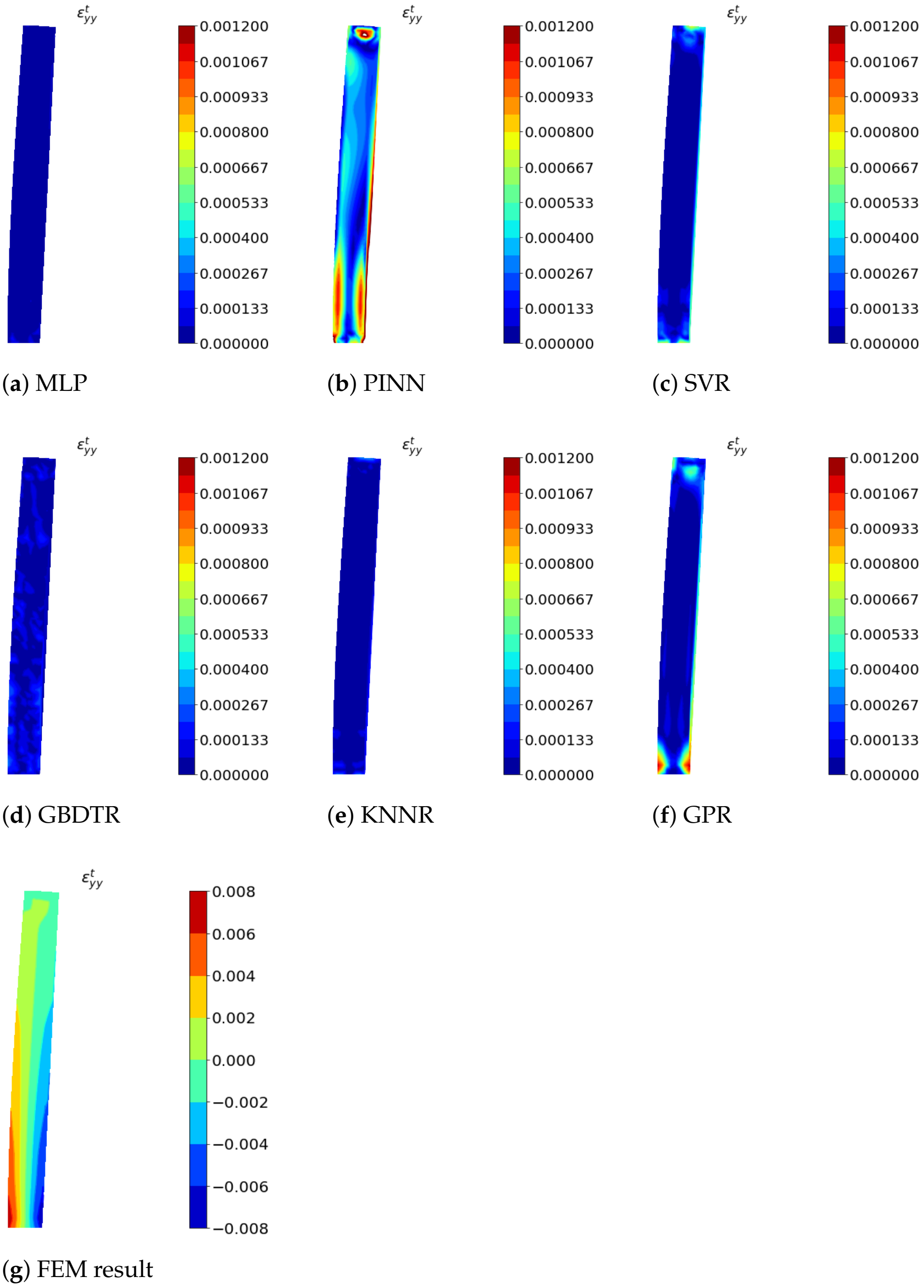
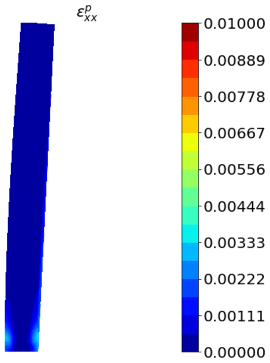
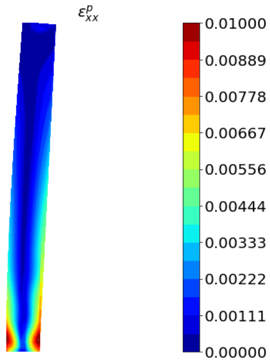


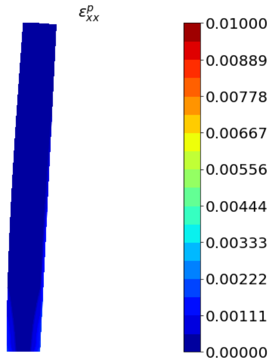
Figure 7. Bending of a beam, Simulation 6 (interpolation): absolute errors of different surrogate models (**a**–**f**) and ground truth Abaqus FEM simulation (**g**) of ε^t_{yy}.



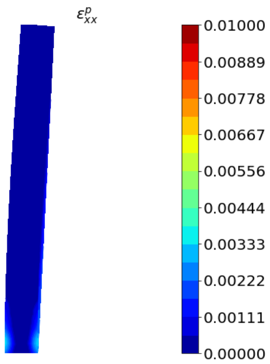
(**a**) MLP



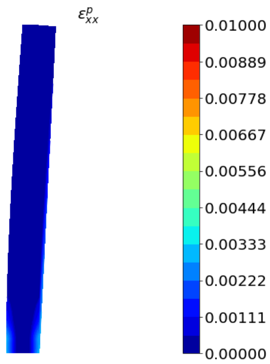
(**b**) PINN



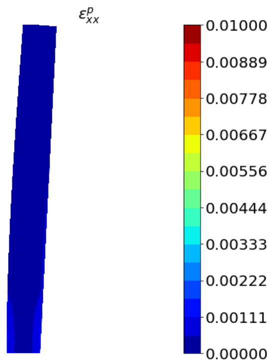
(**c**) SVR



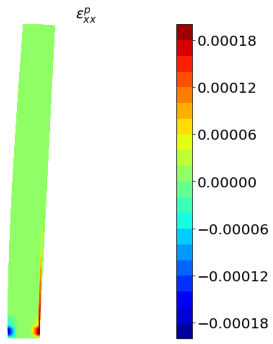
(**d**) GBDTR



(**e**) KNNR



(**f**) GPR



(**g**) FEM result

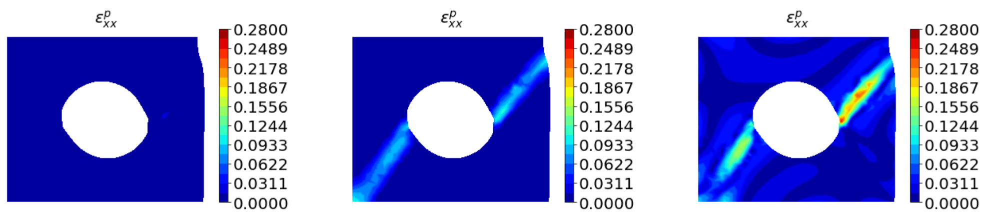
Figure 8. Bending of a beam, Simulation 9 (extrapolation): absolute errors of different surrogate models (**a**–**f**) and ground truth Abaqus FEM simulation (**g**) of ε^p_{xx}.

The compression of a block with four perforations use case presents a more complex setting because we generalize by two generalization variables (yield stress and block width). Therefore, we utilize more training data for this use case; see Table 2. We report the average results of R2-scores with corresponding standard deviations, if applicable, in Table 9.

Table 9. Block: averaged results, bold values indicate the best performing surrogate models. Values in parentheses are the corresponding standard deviations of the average R2-scores, due to repeated experiments of stochastic process models. For further information concerning simulations, see Table 2.

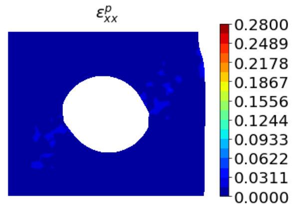
Model	MLP	PINN	SVR	GBDTR	KNNR	GPR	FEM
	Simulation 1						
R2	0.5562 (1.952×10^{-1})	-0.4441 (6.066×10^{-1})	-0.2463	0.5695 (1.665×10^{-3})	**0.7808**	0.1059	-
Inference time [s]	0.0781	0.0638	1.20	0.271	0.00734	0.139	9.08
	Simulation 2						
R2	0.3768 (3.803×10^{-1})	0.1850 (5.531×10^{-2})	-0.1800	0.5149 (3.320×10^{-4})	**0.7366**	0.1409	-
Inference time [s]	0.0457	0.110	0.418	0.0541	0.0790	0.221	6.81
	Simulation 7						
R2	**0.9976** (8.258×10^{-5})	0.9410 (4.310×10^{-3})	0.6415	0.9702 (1.884×10^{-2})	0.9767	0.5200	-
Inference time [s]	0.0442	0.0668	0.402	0.0569	0.0705	0.212	6.61
	Simulation 12						
R2	0.6303 (3.204×10^{-1})	-0.4480 (6.652×10^{-1})	-0.2230	0.5702 (6.266×10^{-4})	**0.7797**	0.1553	-
Inference time [s]	0.0442	0.0668	0.402	0.0569	0.0705	0.212	6.61
	Simulation 13						
R2	0.6475 (3.326×10^{-1})	-0.1122 (3.265×10^{-1})	-0.0745	0.5894 (1.579×10^{-4})	**0.7687**	0.1771	-
Inference time [s]	0.0781	0.0638	1.20	0.271	0.00734	0.139	9.08

As an instance for interpolation, the absolute errors regarding ε^p_{xx} can be seen in Figure 9a–f with Abaqus FEM simulation result (g). Respectively, an instance for extrapolation is shown in Figure 10 with absolute errors (a–f) and FEM ground truth (g). Some models show higher prediction errors near shear bands (high ε^p_{xx} regions) regarding the interpolation task. However, SVR and GPR cannot extract meaningful information from the training data, especially in the space free of plastic deformation. This is indicated by the low average R2-scores, compared to the other models. Considering absolute errors of σ_{xy} and extrapolation the MLP, which is otherwise performing well, shows weaknesses and is in general outperformed by the KNNR.

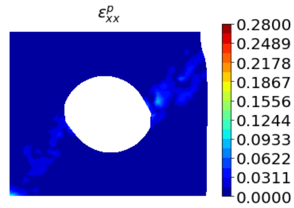


(**a**) MLP (**b**) PINN (**c**) SVR

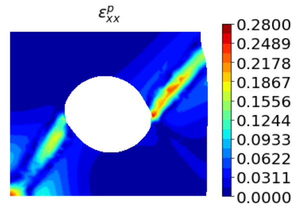
Figure 9. *Cont.*



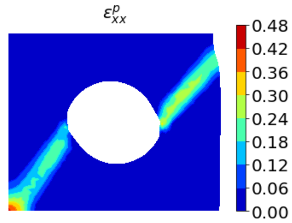
(**d**) GBDTR



(**e**) KNNR

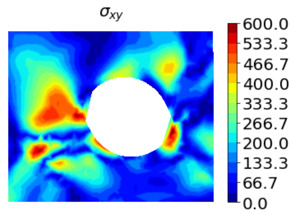


(**f**) GPR

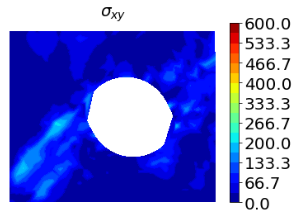


(**g**) FEM result

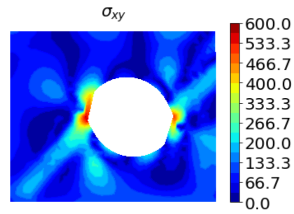
Figure 9. Compression of a block, Simulation 7 (interpolation): absolute errors of different surrogate models (**a**–**f**) and ground truth Abaqus FEM simulation (**g**) of ε^p_{xx}.



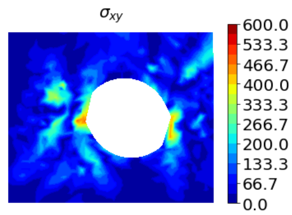
(**a**) MLP



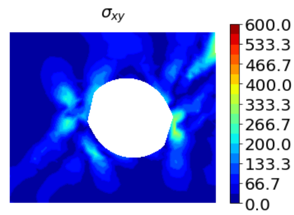
(**b**) PINN



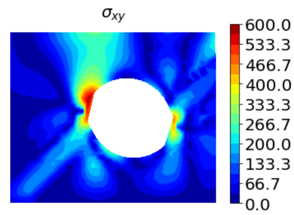
(**c**) SVR



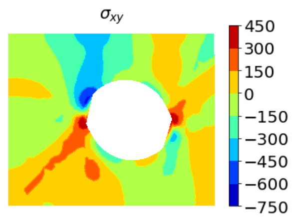
(**d**) GBDTR



(**e**) KNNR



(**f**) GPR



(**g**) FEM result

Figure 10. Compression of a block, Simulation 13 (extrapolation): absolute errors of different surrogate models (**a**–**f**) and ground truth Abaqus FEM simulation (**g**) of σ_{xy}.

4. Discussion

All classes of surrogate models that we considered in this work share several key characteristics: (1) they are mesh-free and thus, can deliver results with infinite resolution; (2) the computation time required to obtain the target values at predefined positions is orders of magnitude lower than for FEM simulations; (3) since for each simulation setup, where the geometry changes, a different mesh is created during FEM simulations, our results indicate that all classes of surrogate models generalize (interpolate) reasonably well across training data positions; (4) furthermore, all surrogate model classes generalize at least to some extent across use case parameters, such as changes in geometry or material parameters. Finally, all surrogate model classes must be used with care, as they do not extrapolate well to data positions and/or use case parameters unseen during training. Our findings show this in the extrapolation result of the beam use case, Simulation 9: due to the greater yield stress, almost no plastic deformation occurs; thus, the surrogate models are not able to learn such material behavior. Similar findings can be seen from the extrapolation results of the block use case, Simulation 1, 2, 12 and 13: approaches utilizing PINNs and SVRs are not able to predict acceptable strain components, leading to overall worse averaged R2-scores. In general, it can be stated that the surrogate models used show similar behavior with respect to inter- and extrapolation, but differ with respect to individual components, i.e., some models are better at predicting individual components (e.g., strains) for unknown generalization variables (e.g., yield strength) than others. Another example would be the symmetric nature of the use case, making it redundant to evaluate, e.g., stresses at negative x-positions, the proposed surrogate models will certainly respond with such stress values, which consequently, cannot be considered meaningful. Similarly, while the surrogate models may well be evaluated at physically meaningless use case parameters, e.g., negative radii, the thus obtained results must be considered meaningless as well. Therefore, all surrogate models must be treated with this in mind, which is a fundamental difference to FEM simulations that do not offer such modes of failure. With these considerations in mind, we now turn to discuss specific characteristics of each surrogate model class.

Our KNNR approach, which can be considered simple compared to the other algorithms, gave competitive results; moreover, this approach showed the best results regarding extrapolation (i.e., Simulations 1, 2, 12 and 13) in the block use case.

Algorithms we constructed with MultiOutputRegressor (SVR and GBDTR) could give better results if the hyperparameters are tuned to each target separately. However, we did not do this for fairness reasons since our other algorithms are also fitted to the overall use case and not to each target individually. We intend to monitor this in the future.

In our setting, the GPR algorithm did not deliver good results. Tuning the kernel function could deliver better results; however, we do not believe that it would be practical to modify for each new simulation use case. Thus, we not intend to head in this direction. However, we plan to investigate whether other Bayesian methods (e.g., Bayesian neural network [30] or neural processes [31]) could be beneficial.

Our MLPs approaches delivered the overall best results in our comparison, especially regarding interpolation (i.e., in the plate and beam use cases Simulations 4 and 6 and in the block use case, Simulation 7). They achieved high accuracies (R2-score > 0.992), while reducing the inference time by a factor of over 100 in comparison to FEM simulations. As mentioned before, designing the architecture is not a straightforward process; however, if the network is deep enough and suitable optimization methods are available (e.g., Adam optimizer) the network can be also efficiently trained utilizing early stopping.

As already reported in literature [32–35], we experienced in our setting that PINNs are not straightforward to design and train. Due to several plateaus in the loss function, early stopping did not prove to be effective. Therefore, we set a fixed number of training epochs. One reason for our observation could be the existence of a non-convex Pareto frontier [36]. In the multi-objective optimization problem, the optimizer might attempt to adjust the model parameters while situated between the different losses, leading it to favor one loss at the expense of the other [37]. Possible approaches to overcome this problem are adaptive optimizers [38], adaptive loss [39], and adaptive activation functions [40]. Moreover, PINNs are objects of current research and will gain more and more attention in the future. Besides other fundamental methods, we additionally plan to aim in that direction for improved surrogate modeling.

5. Conclusions

In this work, we deliver a comprehensive evaluation of generalizable and mesh-free ML and DL surrogate models based on FEM simulation and show that surrogate modeling leads to fast predictions with infinite resolution for practical use. In the context of our evaluation, we show which ML and DL models are target oriented at which level of complexity with respect to prediction accuracy and inference time, which can serve as a basis for the practical implementation of surrogate models (in, for example, production for real-time prediction, cyber–physical systems, and process design).

In future work, we plan to conduct more complex experiments, e.g., generalizing across more input variables regarding geometry (e.g., consideration of all component dimensions) and material parameters (e.g., non-perfect nonlinear material behavior, time-dependent material properties, grain growth, and phase transformation). We will moreover explore extended surrogate models with more complex output variables (e.g., grain size, grain structure, and phase transformation).

Author Contributions: Conceptualization, J.G.H. and P.O.; methodology, J.G.H. and B.C.G.; software, J.G.H.; validation, J.G.H., B.C.G. and P.O.; formal analysis, J.G.H.; investigation, J.G.H.; resources, J.G.H., B.C.G., P.O. and R.K.; data curation, J.G.H.; writing—original draft preparation, J.G.H.; writing—review and editing, J.G.H., B.C.G., P.O. and R.K.; visualization, J.G.H.; supervision, R.K.; project administration, J.G.H. and P.O.; funding acquisition, J.G.H., P.O. and R.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Österreichische Forschungsförderungsgesellschaft (FFG) Grant No. 881039.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The project BrAIN—Brownfield Artificial Intelligence Network for Forging of High Quality Aerospace Components (FFG Grant No. 881039) is funded in the framework of the program 'TAKE OFF', which is a research and technology program of the Austrian Federal Ministry of Transport, Innovation and Technology. The Know-Center is funded within the Austrian COMET Program—Competence Centers for Excellent Technologies—under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry of Economy, Family and Youth and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG. The authors would also like to thank the developers of the sciann API for making their work available and for responding promptly to our questions.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Surrogate Models

We follow the notation introduced in Section 2.2 with data instance $X_i = (x_i, y_i)$ containing input vector x_i and output vector y_i, the number of training instances is n and the number of test instances is m. Notations regarding individual models are introduced when needed.

Appendix A.1. GBDTR

Boosting methods are powerful techniques in which the final "strong" regressor model is based on an iteratively formed ensemble of "weak" base regressor models [41]. The main idea behind boosting is to sequentially add new models to the ensemble, iteratively refining the output. In GBDTR models, boosting is applied to arbitrary differentiable loss functions. In general, GBDTR models are additive models, where the samples are modified so that the labels are set to the negative gradient, while the distribution is held constant [42].

The additive method of GBDTR is the following:

$$\hat{y}_i = F_G(x_i) = \sum_{g=1}^{G} h_g(x_i) \tag{A1}$$

where $\hat{y}_i$ is the prediction for a given input x_i, and h_g are the fitted base tree regressors. The constant G is the number of base tree regressors. The GBDTR algorithm is greedy, where a newly added tree regressor h_g is fitted to minimize the loss L_g of the resulting ensemble $F_g = F_{g-1} + h_g$, i.e.,

$$h_g = arg\ \min_h L_g = arg\ \min_h \sum_{i=1}^{n} l(y_i, F_{g-1}(x_i) + h(x_i)) \tag{A2}$$

Here, $l(y_i, F(x_i))$ is defined by the loss parameters, and $h(x_i)$ is the candidate base regressor. With the utilization of a first-order Taylor approximation:

$$l(z) \approx l(a) + (z-a)\frac{\partial l(a)}{\partial a} \tag{A3}$$

where z corresponds to $F_{g-1}(x_i) + h_g(x_i)$ and a corresponds to $F_{g-1}(x_i)$, we can approximate the value of l with the following:

$$l(y_i, F_{g-1}(x_i) + h_g(x_i)) \approx l(y_i, F_{g-1}(x_i)) + h_g(x_i)\left[\frac{\partial l(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{g-1}} \tag{A4}$$

We denote the derivative of the loss with g_i and remove constant terms:

$$h_m \approx arg\ \min_h \sum_{i=1}^{n} h(x_i) g_i \tag{A5}$$

h_m is minimized if $h(x_i)$ is fitted to predict a value proportional to the negative gradient.

Appendix A.2. KNNR

The KNNR algorithm is a relatively simple method mathematically, compared to other algorithms presented here. Here, the model stores all available use cases from the training dataset D and predicts the numerical target $\hat{y}_j$ of a test query instance x_j with $n < j \le (n+m)$ based on a similarity measure (e.g., distance functions). The algorithm computes the distance-weighted average of the numerical targets of the K nearest neighbors of x_j in D [43].

Specifically, we introduce a distance metric d that measures the distance between all training instances x_i with $i \le n$ and a test instance x_j. Next, the training instances are sorted w.r.t. their respective distance in ascending order to the test instance, i.e., there is a permutation π_j of the training indices i such that $d(x_{\pi_j(1)}, x_j) \le d(x_{\pi_j(2)}, x_j) \le \cdots \le d(x_{\pi_j(n)}, x_j)$. Then, the estimate $\hat{y}_j(x_j)$ is given as the following:

$$\hat{y}_j(x_j) = \frac{1}{K}\sum_{i=1}^{K} y_{\pi_j(i)} \tag{A6}$$

where K must be specified as a hyperparameter.

Appendix A.3. GPR

Gaussian process regression modeling is a non-parametric Bayesian approach [44]. In general, a Gaussian process is a generalization of the Gaussian distribution. The Gaussian distribution describes random variables or random vectors, while a Gaussian process describes a function $f(x)$ [45].

In general, a Gaussian process is completely specified by its mean function $\mu(x)$ and covariance function $K(x, x')$ (also called kernel).

If the function $f(x)$ under consideration is modeled by a Gaussian process, i.e., if $f(x) \sim \mathcal{GP}(\mu(x), K(x, x'))$, then we have the following

$$\mathbb{E}[f(x)] = \mu(x) \tag{A7}$$

$$\mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))] = K(x, x') \tag{A8}$$

for all x and x'. Thus, we can define the Gaussian process as the following:

$$f(x) \sim \mathcal{N}(\mu(x), K(x, x)) \tag{A9}$$

We use the notation that matrix $D = (X_D, Y_D)$ contains the training data with input data matrix $X_D = (x_1, \ldots, x_n)$ and output data matrix $Y_D = (y_1, \ldots, y_n)$, and test data matrix $T = (X_T, Y_T)$ contains the test data with $X_T = (x_{n+1}, \ldots, x_{n+m})$ as input and $Y_T = (y_{n+1}, \ldots, y_{n+m})$ as output. We can define that they are jointly Gaussian and zero mean with consideration of the prior distribution:

$$\begin{bmatrix} Y_D \\ Y_T \end{bmatrix} \sim \mathcal{N}(0, \begin{bmatrix} K(X_D, X_D)) & K(X_D, X_T)) \\ K(X_T, X_D)) & K(X_T, X_T)) \end{bmatrix}) \tag{A10}$$

The Gaussian process makes a prediction Y_T for X_T in a probabilistic way, where, as stated before, the posterior distribution can be fully described by the mean and the covariance.

$$\begin{aligned} Y_T | X_T, X_D, Y_D \sim \mathcal{N}(&K(X_T, X_D)K(X_D, X_D)^{-1}Y_D, \\ &K(X_T, X_T) - K(X_T, X_D)K(X_D, X_D)^{-1}K(X_D, X_T)) \end{aligned} \tag{A11}$$

Appendix A.4. SVR

The SVR approach is a generalization of the SVM classification problem by introducing an ϵ-sensitive region around the approximated function, also called an ϵ-tube. The optimization task in SVR contains two steps: first, finding a convex ϵ-insensitive loss function that need to be minimized, and second, finding the smallest ϵ-tube that contains the most training instances.

The convex optimization has a unique solution and is solved using numerical optimization algorithms. One of the main advantages of SVR is that the computational complexity does not depend on the dimensionality of the input space [46]. To deal with otherwise intractable constraints of the optimization problem, we introduce slack variables ξ_i and ξ_i^* [47]. The positive constant C determines the trade-off between the flatness of the function and the magnitude up to which deviations greater than ϵ are allowed. The primal quadratic optimization problem of SVR is defined as the following:

$$\underset{\omega, b}{\text{minimize}} \; \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{A12}$$

$$subject\ to\ the\ following: \begin{cases} y_i - \omega^T x_i - b & \leq & \epsilon + \xi_i \\ \omega^T x_i + b - y_i & \leq & \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq & 0 \end{cases} \tag{A13}$$

Here, ω is the weight and b the bias to be adjusted. The constrained quadratic optimization problem can be solved by minimizing the Lagrangian with non-negative Lagrange multipliers $\lambda_i, \lambda_i^*, \alpha_i, \alpha_i^*, i \in \{1, \ldots, n\}$:

$$\begin{aligned} \mathcal{L}(\omega, \xi^*, \xi, \lambda, \lambda^*, \alpha, \alpha^*) = \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}\xi_i + \xi_i^* + \sum_{i=1}^{n}\alpha_i^*(y_i - \omega^T x_i - \varepsilon - \xi_i^*) \\ + \sum_{i=1}^{n}\alpha_i(-y_i + \omega^T x_i - \varepsilon - \xi_i) - \sum_{i=1}^{n}\lambda_i\xi_i + \lambda_i^*\xi_i^* \end{aligned} \tag{A14}$$

The minimum of $\mathcal{L}$ can be found by taking the partial derivatives with respect to the variables and making them equal to zero (*Karush-Kuhn-Tucker* (KKT) conditions). With the final KKT condition, we can state the following:

$$\begin{aligned} \alpha_i(-y_i + \omega^T x_i - \varepsilon - \xi_i) &= 0 \\ \alpha_i^*(-y_i + \omega^T x_i - \varepsilon - \xi_i^*) &= 0 \\ \lambda_i \xi_i &= 0 \\ \lambda_i^* \xi_i^* &= 0 \end{aligned} \tag{A15}$$

The Lagrange multipliers that are zero correspond to the inside of the ε-tube, while the support vectors have non-zero Lagrange multipliers. The function estimate depends only on the support vectors, hence this representation is sparse. More specifically, we can derive the following function approximation to predict $\hat{y}_j(x_j)$:

$$\hat{y}_j(x_j) = \sum_{i=1}^{n_{SV}} (\alpha_i^* - \alpha_i) x_i^T x_j \tag{A16}$$

with $\alpha_i, \alpha_i^* \in [0, C]$ and the number of support vectors n_{SV}. For nonlinear SVR we replace $\omega^T x_i$ in (12)–(15) by $\omega^T \phi(x_i)$ and the inner product in (16) by the kernel $K(x_i, x_j)$.

Appendix A.5. MLP

A neural network is a network of simple processing elements, also called neurons. The neurons are arranged in layers. In a fully-connected multi-layer network, a neuron in one layer is connected to every neuron in the layer before and after it. The number of neurons in the input layer is the number of input features p and the number of neurons in the output layer is the number of targets q [48]. MLPs have several theoretical advantages, compared to other ML algorithms. Due to the universal approximation theorem, an MLP can approximate any function if the activation functions of the network are appropriate [49–51]. The MLP makes no prior assumptions about the data distribution, and in many cases, can be trained to generalize to new data not yet seen [52]. However, finding the right architecture and finding the setting of training parameters is not straightforward and usually done by trial and error influenced by the literature and guidelines.

A neural network output $\hat{y}$ corresponding to an input x can be represented as a composition of functions, where the output of layer $L-1$ acts as input to the following layer L. For example, for non-linear activation function σ_L, weight matrix W_L, and bias vector b_L of the respective layer L, we obtain the following:

$$\hat{y}(x) = t_L(x) = \sigma_L(W_L^T t_{L-1}(x) + b_L) \tag{A17}$$

With the neural network estimate $\hat{y}(x)$ and the respective target y of an input x, we can denote a loss function $\mathcal{L}$. A very common loss function for MLPs for regression tasks is the mean-squared error:

$$\mathcal{L}(W, b) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}(x_i) - y_i)^2 \tag{A18}$$

where W and b are the collections of all weight matrices and bias terms, respectively. Optimal weight W^* and bias b^* terms for each layer are identified with minimizing the loss function $\mathcal{L}$ via back-propagation [53].

$$W^*/b^* = \underset{W,b}{\operatorname{argmin}}\ \mathcal{L}(W, b) \tag{A19}$$

Appendix A.6. PINN

In PINNs, the network is trained simultaneously on data and governing differential equations. PINNs are regularized such that their function approximation $\hat{y}(x)$ obeys known laws of physics that apply to the observed data. This type of network is well suited for solving and inverting equations that control physical systems and find application in fluid and solid mechanics as well as in dynamical systems [21,35].

PINNs share similarities with common ANNs, but the loss function has an additional part that describes the physics behind the use case setting. More specifically, the loss $\mathcal{L}$ is composed of the data-driven loss $\mathcal{L}_{data}$ and the physics-informed loss $\mathcal{L}_{physics}$:

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{physics} \tag{A20}$$

While the data-driven loss is often a standard mean-squared error, the physics-informed loss accounts for the degree to which the function approximation solves a given system of governing differential equations. For further details, we refer the reader to [23,35,54] in general and to the Python package of [21,22] in particular for simple implementation of structural mechanics use cases.

Appendix B. Hyperparameters

Table A1. Best performing hyperparameters GBDTR.

	Plate	**Beam**	**Block**
loss	ls	ls	ls
criterion	friedman_mse	mse	friedman_mse
max_features	auto	log2	auto
n_estimators	400	1000	2000

Table A2. Best performing hyperparameters KNNR.

	Plate	**Beam**	**Block**
n_neighbors (K)	7	5	10
weights	distance	distance	distance
algorithm	brute	ball_tree	auto
leaf_size	1	5	1
p_value	1	2	5

Table A3. Best performing hyperparameters GPR.

	Plate	**Beam**	**Block**
kernel	Matern()**2	RationalQuadratic()**2	RationalQuadratic()**2
alpha	10^{-13}	10^{-13}	10^{-14}

Table A4. Best performing hyperparameters SVR.

	Plate	**Beam**	**Block**
kernel	rbf	rbf	rbf
gamma	scale	scale	scale
epsilon	0.005	0.005	0.4
C	95	5	105

Table A5. Best performing hyperparameters MLP.

	Plate	**Beam**	**Block**
hidden layers	3	2	4
neurons	100-100-100	100-100	100-100-100-100
activation function	relu	relu	relu
batch size	32	32	64
validation split	0.1	0.1	0.1
early stopping patience	5000	5000	7500
max epochs	100,000	100,000	100,000
stopped at	27,693	26,383	43,272

Table A6. Best hyperparameters PINN.

	Plate	**Beam**	**Block**
hidden layers	4	4	4
neurons	100-100-100-100	100-100-100-100	100-100-100-100
activation function	tanh	tanh	tanh
batch size	64	64	64
epochs	50,000	50,000	50,000

Appendix C. Detailed Results

Table A7. Detailed results for the plate elongation use case Simulation 1.

		SIMULATION 1								
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.9923	3.121×10^{-6}	0.8331	5.206×10^{-2}	4.117×10^{-1}	5.296×10^{-1}	1.788×10^{-6}	7.973×10^{-1}	5.936×10^{-1}
	ε^t_{xy}	0.9900	3.681×10^{-7}	0.4748	1.814×10^{-1}	5.390×10^{-2}	3.846×10^{-1}	2.065×10^{-5}	5.121×10^{-1}	5.039×10^{-1}
	ε^t_{yy}	0.9924	3.055×10^{-6}	0.8749	9.156×10^{-2}	4.169×10^{-1}	5.281×10^{-1}	1.243×10^{-7}	7.992×10^{-1}	5.950×10^{-1}
	ε^p_{xx}	0.9923	3.269×10^{-6}	0.7385	3.795×10^{-2}	4.079×10^{-1}	5.120×10^{-1}	2.215×10^{-7}	7.964×10^{-1}	5.933×10^{-1}
	ε^p_{xy}	0.9901	2.085×10^{-6}	0.6195	3.218×10^{-1}	4.889×10^{-2}	3.509×10^{-1}	1.003×10^{-5}	5.014×10^{-1}	5.011×10^{-1}
	ε^p_{yy}	0.9923	3.243×10^{-6}	0.7463	3.195×10^{-2}	4.127×10^{-1}	5.069×10^{-1}	5.126×10^{-8}	7.976×10^{-1}	5.941×10^{-1}
	ε^p_{zz}	0.9865	5.307×10^{-6}	0.3235	4.347×10^{-1}	8.349×10^{-1}	7.773×10^{-1}	7.044×10^{-10}	9.194×10^{-1}	6.734×10^{-1}
	σ_{xx}	0.9798	9.128×10^{-6}	0.9496	1.676×10^{-2}	8.886×10^{-1}	7.682×10^{-1}	1.685×10^{-10}	8.854×10^{-1}	7.017×10^{-1}
	σ_{xy}	0.9760	2.915×10^{-7}	0.8405	9.858×10^{-2}	8.373×10^{-1}	6.984×10^{-1}	3.676×10^{-9}	8.605×10^{-1}	6.706×10^{-1}
	σ_{yy}	0.9908	2.639×10^{-6}	0.8574	6.011×10^{-2}	9.822×10^{-1}	8.925×10^{-1}	2.420×10^{-9}	9.120×10^{-1}	5.432×10^{-1}
	σ_{zz}	0.9914	2.684×10^{-6}	0.9484	1.407×10^{-2}	9.208×10^{-1}	8.774×10^{-1}	2.448×10^{-8}	9.326×10^{-1}	6.558×10^{-1}
	u	0.9981	3.358×10^{-8}	0.9690	1.919×10^{-2}	9.095×10^{-1}	8.721×10^{-1}	2.230×10^{-7}	9.443×10^{-1}	6.629×10^{-1}
	v	0.9976	5.954×10^{-9}	0.9610	3.203×10^{-2}	9.195×10^{-1}	8.903×10^{-1}	1.258×10^{-11}	9.549×10^{-1}	6.810×10^{-1}
mean		0.9900	6.155×10^{-9}	0.7797	8.709×10^{-2}	0.6188	0.6606	3.959×10^{-8}	0.8164	0.6131
MSE	ε^t_{xx}	2.916×10^{-5}	4.483×10^{-11}	6.325×10^{-4}	1.973×10^{-4}	2.230×10^{-3}	1.783×10^{-3}	2.569×10^{-11}	7.683×10^{-4}	1.540×10^{-3}
	ε^t_{xy}	3.237×10^{-5}	3.865×10^{-12}	1.702×10^{-3}	5.879×10^{-4}	3.066×10^{-3}	1.994×10^{-3}	2.168×10^{-10}	1.581×10^{-3}	1.608×10^{-3}
	ε^t_{yy}	2.927×10^{-5}	4.523×10^{-11}	4.815×10^{-4}	3.523×10^{-4}	2.244×10^{-3}	1.816×10^{-3}	1.841×10^{-12}	7.727×10^{-4}	1.558×10^{-3}
	ε^p_{xx}	2.945×10^{-5}	4.752×10^{-11}	9.969×10^{-4}	1.447×10^{-4}	2.257×10^{-3}	1.861×10^{-3}	3.220×10^{-12}	7.764×10^{-4}	1.551×10^{-3}
	ε^p_{xy}	2.974×10^{-5}	1.886×10^{-11}	1.144×10^{-3}	9.679×10^{-4}	2.861×10^{-3}	1.952×10^{-3}	9.069×10^{-11}	1.499×10^{-3}	1.500×10^{-3}
	ε^p_{yy}	2.954×10^{-5}	4.803×10^{-11}	9.764×10^{-4}	1.230×10^{-4}	2.260×10^{-3}	1.898×10^{-3}	7.593×10^{-13}	7.789×10^{-4}	1.562×10^{-3}
	ε^p_{zz}	1.921×10^{-9}	1.071×10^{-19}	2.604×10^{-3}	1.673×10^{-3}	2.345×10^{-8}	3.163×10^{-8}	1.421×10^{-23}	1.145×10^{-8}	4.638×10^{-8}
	σ_{xx}	1.599×10^{2}	5.736×10^{2}	3.997×10^{2}	1.329×10^{2}	8.831×10^{2}	1.837×10^{3}	1.059×10^{-2}	9.082×10^{2}	2.364×10^{3}
	σ_{xy}	1.196×10^{2}	7.221	7.941×10^{2}	4.906×10^{2}	8.099×10^{2}	1.501×10^{3}	9.107×10^{-2}	6.941×10^{2}	1.640×10^{3}
	σ_{yy}	5.013×10^{2}	7.911×10^{3}	7.809×10^{3}	3.291×10^{3}	9.766×10^{2}	5.883×10^{3}	7.253	4.818×10^{3}	2.501×10^{4}
	σ_{zz}	1.896×10^{2}	1.297×10^{3}	1.135×10^{3}	3.094×10^{2}	1.741×10^{3}	2.696×10^{3}	1.183×10^{1}	1.481×10^{3}	7.567×10^{3}
	u	4.736×10^{-3}	2.001×10^{-7}	7.558×10^{-2}	4.684×10^{-2}	2.210×10^{-1}	3.123×10^{-1}	1.329×10^{-6}	1.361×10^{-1}	8.230×10^{-1}
	v	0.0055	3.012×10^{-8}	0.0877	7.203×10^{-2}	1.810×10^{-1}	2.467×10^{-1}	6.361×10^{-11}	1.015×10^{-1}	7.174×10^{-1}

Table A8. Detailed results for the plate elongation use case Simulation 4.

SIMULATION 4										
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.9994	1.890×10^{-5}	0.9615	1.029×10^{-2}	0.6110	0.9061	8.599×10^{-2}	0.9354	0.8478
	ε^t_{xy}	0.9984	1.387×10^{-4}	0.6064	2.507×10^{-1}	0.1553	0.8252	1.696×10^{-1}	0.7558	0.6991
	ε^t_{yy}	0.9994	1.450×10^{-5}	0.9817	8.052×10^{-3}	0.6169	0.9067	9.160×10^{-2}	0.9361	0.8495
	ε^p_{xx}	0.9994	1.634×10^{-5}	0.8183	5.486×10^{-2}	0.6087	0.8379	3.180×10^{-2}	0.9346	0.8457
	ε^p_{xy}	0.9984	1.217×10^{-5}	0.9410	7.109×10^{-3}	0.1468	0.7309	1.302×10^{-1}	0.7502	0.6967
	ε^p_{yy}	0.9994	1.572×10^{-5}	0.8881	8.563×10^{-4}	0.6117	0.8487	3.739×10^{-2}	0.9351	0.8467
	ε^p_{zz}	0.9934	1.400×10^{-3}	0.7888	2.487×10^{-3}	0.9349	0.9317	1.525×10^{-2}	0.9790	0.9440
	σ_{xx}	0.9957	1.192×10^{-4}	0.9903	3.579×10^{-4}	0.9326	0.9572	4.072×10^{-2}	0.9742	0.9404
	σ_{xy}	0.9930	6.390×10^{-4}	0.9753	1.977×10^{-4}	0.8643	0.8846	1.103×10^{-1}	0.9417	0.8974
	σ_{yy}	0.9985	9.487×10^{-5}	0.8972	3.716×10^{-3}	0.9932	0.9660	3.214×10^{-2}	0.9909	0.9736
	σ_{zz}	0.9972	1.784×10^{-4}	0.9813	2.062×10^{-4}	0.9813	0.9726	2.625×10^{-2}	0.9901	0.9667
	u	0.9995	4.181×10^{-5}	0.9934	3.368×10^{-4}	0.9297	0.9710	2.879×10^{-2}	0.9792	0.9370
	v	0.9997	6.148×10^{-5}	0.9927	1.157×10^{-3}	0.9392	0.9792	2.026×10^{-2}	0.9857	0.9454
mean		0.9978	1.970×10^{-4}	0.9089	2.598×10^{-2}	0.7174	0.9014	6.310×10^{-2}	0.9298	0.8761
MSE	ε^t_{xx}	2.150×10^{-6}	6.723×10^{-8}	1.370×10^{-4}	3.662×10^{-5}	1.384×10^{-3}	3.200×10^{-4}	3.200×10^{-4}	2.298×10^{-4}	5.412×10^{-4}
	ε^t_{xy}	4.496×10^{-6}	3.991×10^{-7}	1.132×10^{-3}	7.213×10^{-4}	2.430×10^{-3}	4.955×10^{-4}	4.955×10^{-4}	7.027×10^{-4}	8.656×10^{-4}
	ε^t_{yy}	2.186×10^{-6}	5.247×10^{-8}	6.610×10^{-5}	2.914×10^{-5}	1.386×10^{-3}	3.345×10^{-4}	3.345×10^{-4}	2.312×10^{-4}	5.447×10^{-4}
	ε^p_{xx}	2.173×10^{-6}	5.875×10^{-8}	6.531×10^{-4}	1.972×10^{-4}	1.407×10^{-3}	3.485×10^{-4}	3.484×10^{-4}	2.352×10^{-4}	5.547×10^{-4}
	ε^p_{xy}	4.228×10^{-6}	3.139×10^{-8}	1.522×10^{-4}	1.834×10^{-5}	2.202×10^{-3}	5.152×10^{-4}	5.152×10^{-4}	6.446×10^{-4}	7.827×10^{-4}
	ε^p_{yy}	2.194×10^{-6}	5.701×10^{-8}	4.060×10^{-4}	3.106×10^{-6}	1.409×10^{-3}	3.422×10^{-4}	3.422×10^{-4}	2.356×10^{-4}	5.562×10^{-4}
	ε^p_{zz}	7.663×10^{-10}	1.627×10^{-10}	7.659×10^{-4}	9.020×10^{-6}	7.573×10^{-9}	4.990×10^{-9}	4.726×10^{-9}	2.438×10^{-9}	6.512×10^{-9}
	σ_{xx}	5.480×10^{1}	1.515	1.226×10^{2}	4.547	8.561×10^{2}	5.339×10^{2}	5.272×10^{2}	3.283×10^{2}	7.575×10^{2}
	σ_{xy}	3.438×10^{1}	3.155	1.218×10^{2}	9.760×10^{-1}	6.699×10^{2}	5.597×10^{2}	5.547×10^{2}	2.880×10^{2}	5.066×10^{2}
	σ_{yy}	1.267×10^{2}	8.027	8.700×10^{3}	3.144×10^{2}	5.743×10^{2}	3.081×10^{3}	2.514×10^{3}	7.740×10^{2}	2.236×10^{3}
	σ_{zz}	6.863×10^{1}	4.437	4.654×10^{2}	5.129	4.640×10^{2}	6.858×10^{2}	6.491×10^{2}	2.456×10^{2}	8.285×10^{2}
	u	8.003×10^{-4}	6.664×10^{-5}	1.046×10^{-2}	5.368×10^{-4}	1.120×10^{-1}	4.637×10^{-2}	4.578×10^{-2}	3.320×10^{-2}	1.005×10^{-1}
	v	4.276×10^{-4}	8.849×10^{-5}	1.057×10^{-2}	1.666×10^{-3}	8.756×10^{-2}	2.954×10^{-2}	2.952×10^{-2}	2.062×10^{-2}	7.857×10^{-2}

Table A9. Detailed results for the plate elongation use case Simulation 6.

SIMULATION 6										
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.9958	2.566×10^{-3}	0.9336	1.934×10^{-2}	0.6182	0.8405	1.562×10^{-1}	0.9228	0.8366
	ε^t_{xy}	0.9930	4.453×10^{-3}	0.2877	3.328×10^{-1}	0.1915	0.6924	3.030×10^{-1}	0.7280	0.6696
	ε^t_{yy}	0.9959	2.522×10^{-3}	0.9211	5.104×10^{-2}	0.6238	0.8344	1.648×10^{-1}	0.9238	0.8385
	ε^p_{xx}	0.9958	2.606×10^{-3}	0.5414	3.173×10^{-1}	0.6149	0.7529	8.924×10^{-2}	0.9221	0.8352
	ε^p_{xy}	0.9932	4.276×10^{-3}	0.9131	2.635×10^{-2}	0.1831	0.5966	1.726×10^{-1}	0.7218	0.6668
	ε^p_{yy}	0.9958	2.590×10^{-3}	0.7744	8.943×10^{-2}	0.6178	0.7799	9.094×10^{-2}	0.9228	0.8362
	ε^p_{zz}	0.9901	2.688×10^{-3}	0.8264	2.288×10^{-2}	0.9590	0.8938	2.641×10^{-3}	0.9860	0.9487
	σ_{xx}	0.9837	1.326×10^{-2}	0.9839	3.098×10^{-4}	0.9527	0.9600	3.838×10^{-2}	0.9799	0.9407
	σ_{xy}	0.9768	8.968×10^{-3}	0.9655	2.023×10^{-3}	0.8687	0.8571	1.381×10^{-1}	0.9431	0.9030
	σ_{yy}	0.9890	1.196×10^{-2}	0.9103	1.363×10^{-3}	0.9908	0.9729	2.571×10^{-2}	0.9938	0.9709
	σ_{zz}	0.9900	8.749×10^{-3}	0.9851	1.518×10^{-3}	0.9818	0.9682	3.102×10^{-2}	0.9923	0.9636
	u	0.9980	1.261×10^{-3}	0.9835	1.344×10^{-3}	0.9077	0.9366	6.320×10^{-2}	0.9717	0.9336
	v	0.9988	6.926×10^{-4}	0.9849	6.032×10^{-3}	0.9161	0.9685	3.118×10^{-2}	0.9760	0.9359
mean		0.9920	1.889×10^{-3}	0.8470	6.309×10^{-2}	0.7251	0.8503	1.005×10^{-1}	0.9219	0.8676
MSE	ε^t_{xx}	1.563×10^{-5}	9.386×10^{-6}	2.475×10^{-4}	7.210×10^{-5}	1.423×10^{-3}	5.884×10^{-4}	5.884×10^{-4}	2.878×10^{-4}	6.090×10^{-4}
	ε^t_{xy}	2.402×10^{-5}	1.372×10^{-5}	2.360×10^{-3}	1.103×10^{-3}	2.680×10^{-3}	1.012×10^{-3}	1.012×10^{-3}	9.014×10^{-4}	1.095×10^{-3}
	ε^t_{yy}	1.579×10^{-5}	9.392×10^{-6}	2.994×10^{-4}	1.936×10^{-4}	1.427×10^{-3}	6.268×10^{-4}	6.268×10^{-4}	2.890×10^{-4}	6.128×10^{-4}
	ε^p_{xx}	1.596×10^{-5}	9.635×10^{-6}	1.725×10^{-3}	1.194×10^{-3}	1.449×10^{-3}	6.327×10^{-4}	6.327×10^{-4}	2.929×10^{-4}	6.201×10^{-4}
	ε^p_{xy}	2.135×10^{-5}	1.199×10^{-5}	2.627×10^{-4}	7.965×10^{-5}	2.469×10^{-3}	8.705×10^{-4}	8.705×10^{-4}	8.407×10^{-4}	1.007×10^{-3}
	ε^p_{yy}	1.599×10^{-5}	9.654×10^{-6}	8.559×10^{-4}	3.392×10^{-4}	1.450×10^{-3}	5.899×10^{-4}	5.899×10^{-4}	2.930×10^{-4}	6.214×10^{-4}
	ε^p_{zz}	1.341×10^{-9}	6.361×10^{-10}	6.584×10^{-4}	8.677×10^{-5}	4.377×10^{-9}	5.918×10^{-9}	5.708×10^{-9}	1.492×10^{-9}	5.484×10^{-9}
	σ_{xx}	4.784×10^{2}	5.454×10^{2}	2.162×10^{2}	4.170	6.368×10^{2}	5.300×10^{2}	5.246×10^{2}	2.709×10^{2}	7.981×10^{2}
	σ_{xy}	2.621×10^{2}	2.590×10^{2}	1.620×10^{2}	9.491	6.161×10^{2}	6.614×10^{2}	6.566×10^{2}	2.669×10^{2}	4.552×10^{2}
	σ_{yy}	2.230×10^{3}	2.811×10^{3}	8.488×10^{3}	1.291×10^{2}	8.701×10^{2}	2.741×10^{3}	2.264×10^{3}	5.900×10^{2}	2.755×10^{3}
	σ_{zz}	4.078×10^{2}	4.385×10^{2}	3.842×10^{2}	3.926×10^{1}	4.705×10^{2}	8.263×10^{2}	7.996×10^{2}	1.981×10^{2}	9.411×10^{2}
	u	2.471×10^{-3}	1.295×10^{-3}	1.953×10^{-2}	1.591×10^{-3}	1.093×10^{-1}	7.515×10^{-2}	7.473×10^{-2}	3.349×10^{-2}	7.862×10^{-2}
	v	1.594×10^{-3}	3.341×10^{-4}	1.621×10^{-2}	6.487×10^{-3}	9.022×10^{-2}	3.373×10^{-2}	3.371×10^{-2}	2.578×10^{-2}	6.895×10^{-2}

Table A10. Detailed results for the plate elongation use case Simulation 9.

		MLP		PINN		SVR	GBDTR		KNNR	GPR
		mean	**std**	**mean**	**std**		**mean**	**std**		
					SIMULATION 9					
R2	ε^t_{xx}	0.9902	4.611×10^{-7}	0.8149	1.175×10^{-1}	5.305×10^{-1}	7.066×10^{-1}	4.844×10^{-7}	8.255×10^{-1}	5.599×10^{-1}
	ε^t_{xy}	0.9699	1.932×10^{-5}	0.3356	9.434×10^{-2}	5.673×10^{-2}	3.932×10^{-1}	1.518×10^{-8}	4.667×10^{-1}	3.628×10^{-1}
	ε^t_{yy}	0.9903	4.339×10^{-7}	0.8484	1.445×10^{-1}	5.344×10^{-1}	7.197×10^{-1}	3.018×10^{-8}	8.272×10^{-1}	5.611×10^{-1}
	ε^p_{xx}	0.9904	3.884×10^{-7}	0.6097	7.907×10^{-2}	5.180×10^{-1}	7.195×10^{-1}	4.415×10^{-11}	8.235×10^{-1}	5.571×10^{-1}
	ε^p_{xy}	0.9704	1.921×10^{-5}	0.5937	3.410×10^{-1}	4.617×10^{-2}	4.245×10^{-1}	3.808×10^{-8}	4.609×10^{-1}	3.586×10^{-1}
	ε^p_{yy}	0.9904	3.972×10^{-7}	0.7357	4.504×10^{-2}	5.204×10^{-1}	7.128×10^{-1}	7.477×10^{-7}	8.242×10^{-1}	5.579×10^{-1}
	ε^p_{zz}	0.9628	7.728×10^{-6}	0.3747	4.181×10^{-1}	9.292×10^{-1}	8.309×10^{-1}	6.767×10^{-9}	9.016×10^{-1}	6.839×10^{-1}
	σ_{xx}	0.9633	3.899×10^{-4}	0.9577	1.764×10^{-2}	9.258×10^{-1}	9.264×10^{-1}	1.662×10^{-9}	9.516×10^{-1}	6.847×10^{-1}
	σ_{xy}	0.9438	1.369×10^{-3}	0.8168	1.079×10^{-1}	8.161×10^{-1}	6.292×10^{-1}	1.792×10^{-7}	7.731×10^{-1}	5.730×10^{-1}
	σ_{yy}	0.9876	5.783×10^{-5}	0.9443	4.324×10^{-3}	9.779×10^{-1}	9.280×10^{-1}	1.478×10^{-11}	9.367×10^{-1}	5.502×10^{-1}
	σ_{zz}	0.9865	1.062×10^{-5}	0.9526	2.420×10^{-2}	9.704×10^{-1}	9.328×10^{-1}	3.200×10^{-10}	9.407×10^{-1}	6.108×10^{-1}
	u	0.9898	3.302×10^{-6}	0.9168	6.792×10^{-2}	8.492×10^{-1}	6.756×10^{-1}	2.928×10^{-7}	8.302×10^{-1}	6.322×10^{-1}
	v	0.9859	2.747×10^{-6}	0.9300	6.483×10^{-2}	8.630×10^{-1}	8.433×10^{-1}	2.501×10^{-8}	8.963×10^{-1}	6.544×10^{-1}
mean		0.9786	2.970×10^{-5}	0.7562	1.046×10^{-1}	0.6568	0.7263	9.780×10^{-9}	0.8045	0.5651
MSE	ε^t_{xx}	3.323×10^{-5}	5.335×10^{-12}	6.296×10^{-4}	3.996×10^{-4}	1.597×10^{-3}	9.979×10^{-4}	5.605×10^{-12}	5.937×10^{-4}	1.497×10^{-3}
	ε^t_{xy}	1.069×10^{-4}	2.439×10^{-10}	2.360×10^{-3}	3.352×10^{-4}	3.351×10^{-3}	2.156×10^{-3}	1.916×10^{-13}	1.895×10^{-3}	2.264×10^{-3}
	ε^t_{yy}	3.349×10^{-5}	5.193×10^{-12}	5.245×10^{-4}	5.000×10^{-4}	1.611×10^{-3}	9.698×10^{-4}	3.612×10^{-13}	5.977×10^{-4}	1.518×10^{-3}
	ε^p_{xx}	3.322×10^{-5}	4.646×10^{-12}	1.350×10^{-3}	2.735×10^{-4}	1.667×10^{-3}	9.703×10^{-4}	5.282×10^{-16}	6.106×10^{-4}	1.532×10^{-3}
	ε^p_{xy}	9.449×10^{-5}	1.962×10^{-10}	1.298×10^{-3}	1.090×10^{-3}	3.048×10^{-3}	1.839×10^{-3}	3.889×10^{-13}	1.723×10^{-3}	2.050×10^{-3}
	ε^p_{yy}	3.332×10^{-5}	4.818×10^{-12}	9.206×10^{-4}	1.569×10^{-4}	1.670×10^{-3}	1.000×10^{-3}	9.069×10^{-12}	6.123×10^{-4}	1.540×10^{-3}
	ε^p_{zz}	2.279×10^{-9}	2.901×10^{-20}	2.178×10^{-3}	1.456×10^{-3}	4.339×10^{-9}	1.036×10^{-8}	2.540×10^{-23}	6.026×10^{-9}	1.937×10^{-8}
	σ_{xx}	3.733×10^{2}	4.030×10^{4}	4.303×10^{2}	1.794×10^{2}	7.548×10^{2}	7.480×10^{2}	1.718×10^{-1}	4.916×10^{2}	3.206×10^{3}
	σ_{xy}	1.946×10^{2}	1.639×10^{4}	6.338×10^{2}	3.734×10^{2}	6.364×10^{2}	1.283×10^{3}	2.146	7.852×10^{2}	1.478×10^{3}
	σ_{yy}	1.092×10^{3}	4.456×10^{5}	4.890×10^{3}	3.796×10^{2}	1.943×10^{3}	6.320×10^{3}	1.139×10^{-1}	5.556×10^{3}	3.948×10^{4}
	σ_{zz}	2.613×10^{2}	3.972×10^{3}	9.161×10^{2}	4.681×10^{2}	5.727×10^{2}	1.300×10^{3}	1.197×10^{-1}	1.147×10^{3}	7.528×10^{3}
	u	5.525×10^{-3}	9.688×10^{-7}	4.508×10^{-2}	3.679×10^{-2}	8.165×10^{-2}	1.757×10^{-1}	8.590×10^{-8}	9.196×10^{-2}	1.992×10^{-1}
	v	0.0072	7.221×10^{-7}	0.0359	3.324×10^{-2}	7.021×10^{-2}	8.034×10^{-2}	6.573×10^{-9}	5.317×10^{-2}	1.772×10^{-1}

Table A11. Detailed results for the bending beam use case Simulation 1.

		MLP		PINN		SVR	GBDTR		KNNR	GPR
		mean	**std**	**mean**	**std**		**mean**	**std**		
					SIMULATION 1					
R2	ε^t_{xx}	0.8367	8.066×10^{-5}	0.7682	1.742×10^{-2}	0.8036	0.8566	2.006×10^{-6}	0.8377	0.6790
	ε^t_{xy}	0.8570	6.906×10^{-5}	0.6632	1.499×10^{-1}	0.4932	0.9030	2.784×10^{-7}	0.6409	0.5727
	ε^t_{yy}	0.9594	1.142×10^{-5}	0.8487	3.318×10^{-4}	0.9520	0.9651	2.776×10^{-9}	0.9607	0.8805
	ε^p_{xx}	0.0647	7.879×10^{-6}	0.0304	1.866×10^{-4}	0.0083	0.1599	1.015×10^{-5}	0.1426	0.0633
	ε^p_{xy}	−0.0091	4.606×10^{-4}	−0.0106	3.336×10^{-4}	−0.0051	0.0906	3.080×10^{-6}	0.0098	0.0652
	ε^p_{yy}	0.0723	2.093×10^{-5}	0.0335	4.720×10^{-4}	0.0051	0.1746	9.704×10^{-7}	0.1568	0.0720
	ε^p_{zz}	0.1157	3.506×10^{-4}	−0.0078	8.842×10^{-4}	0.0152	0.2458	1.254×10^{-6}	0.2373	0.1278
	σ_{xx}	0.9643	3.732×10^{-4}	0.9822	2.748×10^{-3}	0.0291	0.9837	3.203×10^{-7}	0.6293	0.1681
	σ_{xy}	0.9157	1.814×10^{-4}	0.8902	1.515×10^{-2}	0.4227	0.9451	4.972×10^{-6}	0.6324	0.5024
	σ_{yy}	0.9482	8.435×10^{-5}	0.9546	1.249×10^{-3}	0.9618	0.9547	1.283×10^{-7}	0.9540	0.9815
	σ_{zz}	0.9789	1.711×10^{-5}	0.9742	1.819×10^{-3}	0.9766	0.9818	2.750×10^{-7}	0.9781	0.9521
	u	0.9948	6.208×10^{-6}	0.9974	5.933×10^{-4}	0.9978	0.9974	1.963×10^{-10}	0.9972	0.9678
	v	0.9875	2.782×10^{-5}	0.8897	6.238×10^{-2}	0.9976	0.9973	6.646×10^{-8}	0.9976	0.9580
mean		0.6682	7.345×10^{-6}	0.6165	1.648×10^{-2}	0.5122	0.7120	1.088×10^{-8}	0.6288	0.5377
MSE	ε^t_{xx}	3.452×10^{-7}	3.602×10^{-16}	4.899×10^{-7}	3.682×10^{-8}	4.151×10^{-7}	3.077×10^{-7}	9.363×10^{-17}	3.430×10^{-7}	6.783×10^{-7}
	ε^t_{xy}	3.723×10^{-8}	4.679×10^{-18}	8.765×10^{-8}	3.903×10^{-8}	1.319×10^{-7}	2.554×10^{-8}	7.242×10^{-20}	9.346×10^{-8}	1.112×10^{-7}
	ε^t_{yy}	2.876×10^{-7}	5.739×10^{-16}	1.073×10^{-6}	2.352×10^{-9}	3.402×10^{-7}	2.484×10^{-7}	3.517×10^{-18}	2.785×10^{-7}	8.473×10^{-7}
	ε^p_{xx}	4.778×10^{-7}	2.056×10^{-18}	4.953×10^{-7}	9.531×10^{-11}	5.066×10^{-7}	4.288×10^{-7}	4.320×10^{-18}	4.380×10^{-7}	4.785×10^{-7}
	ε^p_{xy}	1.779×10^{-8}	1.431×10^{-19}	1.781×10^{-8}	5.879×10^{-12}	1.772×10^{-8}	1.604×10^{-8}	2.456×10^{-21}	1.745×10^{-8}	1.648×10^{-8}
	ε^p_{yy}	6.251×10^{-7}	9.503×10^{-18}	6.512×10^{-7}	3.180×10^{-10}	6.704×10^{-7}	5.577×10^{-7}	2.451×10^{-18}	5.681×10^{-7}	6.253×10^{-7}
	ε^p_{zz}	1.029×10^{-8}	4.752×10^{-20}	6.790×10^{-7}	5.958×10^{-10}	1.146×10^{-8}	8.783×10^{-9}	8.663×10^{-23}	8.878×10^{-9}	1.015×10^{-8}
	σ_{xx}	9.974×10^{1}	2.906×10^{3}	4.964×10^{1}	7.668	2.709×10^{3}	4.422×10^{1}	5.128×10^{-2}	1.035×10^{3}	2.322×10^{3}
	σ_{xy}	7.615×10^{1}	1.479×10^{2}	9.920×10^{1}	1.368×10^{1}	5.214×10^{2}	5.007×10^{1}	7.083	3.320×10^{2}	4.494×10^{2}
	σ_{yy}	1.295×10^{4}	5.275×10^{6}	1.135×10^{4}	3.123×10^{2}	9.543×10^{3}	1.141×10^{4}	4.827×10^{4}	1.150×10^{4}	4.631×10^{3}
	σ_{zz}	5.921×10^{2}	1.342×10^{4}	7.215×10^{2}	5.094×10^{1}	6.548×10^{2}	5.054×10^{2}	6.585×10^{1}	6.130×10^{2}	1.342×10^{3}
	u	1.251×10^{-2}	3.633×10^{-5}	6.323×10^{-3}	1.435×10^{-3}	5.220×10^{-3}	6.289×10^{-3}	1.284×10^{-8}	6.723×10^{-3}	7.797×10^{-2}
	v	4.059×10^{-4}	2.945×10^{-8}	3.589×10^{-3}	2.030×10^{-3}	7.902×10^{-5}	8.486×10^{-5}	2.337×10^{-11}	7.796×10^{-5}	1.367×10^{-3}

Table A12. Detailed results for the bending beam use case Simulation 4.

SIMULATION 4										
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.9992	4.882×10^{-4}	0.9839	6.920×10^{-4}	0.9601	0.9928	1.008×10^{-3}	0.9940	0.9590
	ε^t_{xy}	0.9977	1.512×10^{-3}	0.9643	8.745×10^{-3}	0.5794	0.9941	1.496×10^{-4}	0.9321	0.7219
	ε^t_{yy}	0.9996	2.146×10^{-4}	0.9793	1.865×10^{-4}	0.9970	0.9981	4.564×10^{-4}	0.9995	0.9922
	ε^p_{xx}	0.9965	1.939×10^{-3}	0.8471	6.012×10^{-3}	0.8796	0.8709	2.564×10^{-3}	0.9819	0.8397
	ε^p_{xy}	0.9894	8.291×10^{-3}	0.9911	4.488×10^{-4}	0.0827	0.8513	3.840×10^{-3}	0.7997	0.3642
	ε^p_{yy}	0.9972	1.521×10^{-3}	0.8546	9.650×10^{-4}	0.8968	0.8881	2.980×10^{-3}	0.9855	0.8495
	ε^p_{zz}	0.9985	6.214×10^{-4}	0.7411	1.351×10^{-2}	0.9352	0.9479	2.379×10^{-3}	0.9944	0.8838
	σ_{xx}	0.9981	8.971×10^{-4}	0.9987	5.263×10^{-4}	0.0418	0.9979	7.556×10^{-5}	0.9046	0.4887
	σ_{xy}	0.9974	1.516×10^{-3}	0.9799	1.242×10^{-2}	0.4600	0.9946	4.227×10^{-4}	0.9167	0.6331
	σ_{yy}	0.9997	1.157×10^{-4}	0.9169	2.384×10^{-4}	0.9992	0.9983	1.213×10^{-5}	0.9998	0.9965
	σ_{zz}	0.9997	1.080×10^{-4}	0.9852	2.265×10^{-4}	0.9939	0.9990	1.026×10^{-4}	0.9989	0.9916
	u	0.9998	2.885×10^{-5}	0.9989	3.940×10^{-4}	1.0000	0.9996	7.767×10^{-5}	0.9998	0.9989
	v	0.9997	4.696×10^{-5}	0.9517	1.110×10^{-3}	1.0000	0.9995	4.749×10^{-5}	0.9999	0.9974
mean		0.9979	1.319×10^{-3}	0.9379	1.042×10^{-3}	0.7558	0.9640	6.751×10^{-4}	0.9621	0.8243
MSE	ε^t_{xx}	1.158×10^{-9}	7.461×10^{-10}	2.467×10^{-8}	1.057×10^{-9}	6.097×10^{-8}	1.099×10^{-8}	1.541×10^{-9}	9.198×10^{-9}	6.258×10^{-8}
	ε^t_{xy}	5.680×10^{-10}	3.784×10^{-10}	8.924×10^{-9}	2.188×10^{-9}	1.052×10^{-7}	1.471×10^{-9}	3.743×10^{-11}	1.700×10^{-8}	6.960×10^{-8}
	ε^t_{yy}	2.708×10^{-9}	1.457×10^{-9}	1.404×10^{-7}	1.267×10^{-9}	2.058×10^{-8}	1.267×10^{-8}	3.100×10^{-9}	3.589×10^{-9}	5.309×10^{-8}
	ε^p_{xx}	3.854×10^{-10}	2.155×10^{-10}	1.699×10^{-8}	6.681×10^{-10}	1.337×10^{-8}	1.435×10^{-8}	2.849×10^{-10}	2.012×10^{-9}	1.782×10^{-8}
	ε^p_{xy}	4.304×10^{-11}	3.372×10^{-11}	3.608×10^{-11}	1.825×10^{-12}	3.730×10^{-9}	6.049×10^{-10}	1.562×10^{-11}	8.146×10^{-10}	2.586×10^{-9}
	ε^p_{yy}	4.490×10^{-10}	2.477×10^{-10}	2.367×10^{-8}	1.571×10^{-10}	1.679×10^{-8}	1.822×10^{-8}	4.851×10^{-10}	2.358×10^{-9}	2.449×10^{-8}
	ε^p_{zz}	7.588×10^{-12}	3.101×10^{-12}	4.214×10^{-8}	2.199×10^{-9}	3.233×10^{-10}	2.600×10^{-10}	1.187×10^{-11}	2.771×10^{-11}	5.800×10^{-10}
	σ_{xx}	6.301	2.904	4.242	1.704	3.102×10^{3}	6.781	2.446×10^{-1}	3.087×10^{2}	1.655×10^{3}
	σ_{xy}	2.546	1.489	1.974×10^{1}	1.220×10^{1}	5.306×10^{2}	5.268	4.153×10^{-1}	8.179×10^{1}	3.604×10^{2}
	σ_{yy}	9.601×10^{1}	3.555×10^{1}	2.554×10^{4}	7.326×10^{1}	2.483×10^{2}	5.289×10^{2}	3.727	5.885×10^{1}	1.065×10^{3}
	σ_{zz}	9.587	3.424	4.705×10^{2}	7.179	1.947×10^{2}	3.277×10^{1}	3.251	3.488×10^{1}	2.663×10^{2}
	u	5.949×10^{-4}	7.015×10^{-5}	2.758×10^{-3}	9.579×10^{-4}	1.676×10^{-5}	8.592×10^{-4}	1.889×10^{-4}	3.838×10^{-4}	2.781×10^{-3}
	v	9.229×10^{-6}	1.558×10^{-6}	1.601×10^{-3}	3.683×10^{-5}	5.846×10^{-7}	1.598×10^{-5}	1.576×10^{-6}	2.105×10^{-6}	8.758×10^{-5}

Table A13. Detailed results for the bending beam use case Simulation 6.

SIMULATION 6										
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.9997	1.585×10^{-4}	0.9827	1.679×10^{-3}	0.9606	0.9970	3.028×10^{-4}	0.9946	0.9615
	ε^t_{xy}	0.9988	5.440×10^{-4}	0.9636	1.211×10^{-2}	0.6360	0.9956	2.013×10^{-4}	0.9418	0.7492
	ε^t_{yy}	0.9997	1.581×10^{-4}	0.9683	3.948×10^{-4}	0.9979	0.9990	1.438×10^{-4}	0.9997	0.9935
	ε^p_{xx}	0.9973	1.546×10^{-3}	0.8567	3.014×10^{-3}	0.7713	0.8529	1.868×10^{-3}	0.9825	0.7532
	ε^p_{xy}	0.9887	5.094×10^{-3}	0.9788	7.437×10^{-3}	0.1049	0.7773	4.855×10^{-3}	0.7837	0.3351
	ε^p_{yy}	0.9979	1.115×10^{-3}	0.8769	1.114×10^{-2}	0.7980	0.8627	1.675×10^{-3}	0.9851	0.7650
	ε^p_{zz}	0.9980	7.159×10^{-4}	0.6162	1.176×10^{-2}	0.8392	0.8955	3.934×10^{-4}	0.9910	0.8054
	σ_{xx}	0.9985	5.385×10^{-4}	0.9993	1.296×10^{-4}	0.0374	0.9987	1.248×10^{-4}	0.9051	0.4865
	σ_{xy}	0.9984	7.070×10^{-4}	0.9819	1.416×10^{-2}	0.4890	0.9953	2.116×10^{-4}	0.9198	0.6422
	σ_{yy}	0.9997	1.586×10^{-4}	0.9465	1.587×10^{-3}	0.9993	0.9988	1.469×10^{-4}	0.9999	0.9969
	σ_{zz}	0.9997	1.588×10^{-4}	0.9890	3.975×10^{-5}	0.9937	0.9991	1.271×10^{-4}	0.9990	0.9919
	u	0.9997	1.787×10^{-5}	0.9984	2.302×10^{-4}	1.0000	0.9997	6.919×10^{-5}	0.9998	0.9989
	v	0.9997	6.709×10^{-5}	0.9503	1.090×10^{-3}	1.0000	0.9995	1.477×10^{-4}	0.9999	0.9974
mean		0.9981	8.315×10^{-4}	0.9314	8.396×10^{-4}	0.7406	0.9516	1.368×10^{-4}	0.9617	0.8059
MSE	ε^t_{xx}	4.762×10^{-10}	2.159×10^{-10}	2.352×10^{-8}	2.287×10^{-9}	5.373×10^{-8}	4.027×10^{-9}	4.125×10^{-10}	7.327×10^{-9}	5.246×10^{-8}
	ε^t_{xy}	2.949×10^{-10}	1.330×10^{-10}	8.896×10^{-9}	2.962×10^{-9}	8.902×10^{-8}	1.069×10^{-9}	4.922×10^{-11}	1.423×10^{-8}	6.133×10^{-8}
	ε^t_{yy}	1.818×10^{-9}	1.064×10^{-9}	2.134×10^{-7}	2.658×10^{-9}	1.434×10^{-8}	6.562×10^{-9}	9.683×10^{-10}	2.272×10^{-9}	4.385×10^{-8}
	ε^p_{xx}	9.135×10^{-11}	5.298×10^{-11}	4.910×10^{-9}	1.033×10^{-10}	7.838×10^{-9}	5.041×10^{-9}	6.403×10^{-11}	5.983×10^{-10}	8.458×10^{-9}
	ε^p_{xy}	1.017×10^{-11}	4.605×10^{-12}	1.921×10^{-11}	6.723×10^{-12}	8.092×10^{-10}	2.014×10^{-10}	4.389×10^{-12}	1.955×10^{-10}	6.011×10^{-10}
	ε^p_{yy}	1.126×10^{-10}	5.894×10^{-11}	6.510×10^{-9}	5.889×10^{-10}	1.068×10^{-8}	7.262×10^{-9}	8.856×10^{-11}	7.876×10^{-10}	1.242×10^{-8}
	ε^p_{zz}	4.137×10^{-12}	1.456×10^{-12}	2.030×10^{-8}	6.220×10^{-10}	3.271×10^{-10}	2.126×10^{-10}	8.001×10^{-13}	1.840×10^{-11}	3.959×10^{-10}
	σ_{xx}	4.891	1.801	2.311	4.333×10^{-1}	3.219×10^{3}	4.183	4.173×10^{-1}	3.174×10^{2}	1.717×10^{3}
	σ_{xy}	1.582	7.091×10^{-1}	1.820×10^{1}	1.420×10^{1}	5.125×10^{2}	4.749	2.123×10^{-1}	8.043×10^{1}	3.589×10^{2}
	σ_{yy}	8.720×10^{1}	5.288×10^{1}	1.785×10^{4}	5.289×10^{2}	2.192×10^{2}	4.124×10^{2}	4.896×10^{1}	4.382×10^{1}	1.044×10^{3}
	σ_{zz}	8.678	5.215	3.603×10^{2}	1.305	2.061×10^{2}	2.859×10^{1}	4.174	3.387×10^{1}	2.660×10^{2}
	u	7.648×10^{-4}	4.353×10^{-5}	3.887×10^{-3}	5.607×10^{-4}	1.611×10^{-5}	6.987×10^{-4}	1.685×10^{-4}	3.679×10^{-4}	2.779×10^{-3}
	v	9.476×10^{-6}	2.243×10^{-6}	1.663×10^{-3}	3.645×10^{-5}	4.862×10^{-7}	1.622×10^{-5}	4.937×10^{-6}	1.937×10^{-6}	8.695×10^{-5}

Table A14. Detailed results for the bending beam use case Simulation 9.

		SIMULATION 9								
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.7851	3.018×10^{-3}	0.6358	6.166×10^{-4}	0.8656	0.8172	3.255×10^{-5}	0.7939	0.9395
	ε^t_{xy}	0.8255	4.803×10^{-3}	0.8392	2.690×10^{-2}	0.6478	0.9133	4.185×10^{-7}	0.7931	0.6226
	ε^t_{yy}	0.9627	3.529×10^{-6}	0.9126	3.213×10^{-4}	0.9776	0.9721	3.246×10^{-8}	0.9732	0.9432
	ε^p_{xx}	−984.3572	3.895×10^{4}	−2372.9520	1.322×10^{1}	−305.8183	−823.3812	3.067×10^{1}	-809.9266	−220.5149
	ε^p_{xy}	−13394.3542	1.990×10^{6}	−13885.9308	3.975×10^{2}	−604.7674	−8965.1296	4.317×10^{2}	−2397.3474	−4955.1166
	ε^p_{yy}	−821.5068	2.422×10^{4}	−343.2822	2.128	−282.6090	−698.7968	1.881×10^{-1}	−683.0748	−193.7274
	ε^p_{zz}	−359.9536	1.372×10^{3}	−382.8576	1.639×10^{1}	−214.6722	−322.9159	4.464×10^{-2}	−309.8423	−109.3652
	σ_{xx}	0.9529	1.611×10^{-3}	0.9893	1.808×10^{-3}	0.0275	0.9932	2.982×10^{-9}	0.5629	0.2630
	σ_{xy}	0.9120	1.109×10^{-3}	0.9260	1.163×10^{-2}	0.4862	0.9627	3.386×10^{-7}	0.6904	0.5129
	σ_{yy}	0.9672	5.183×10^{-6}	0.8105	3.138×10^{-3}	0.9569	0.9763	4.897×10^{-8}	0.9745	0.8841
	σ_{zz}	0.9825	1.315×10^{-5}	0.9681	8.932×10^{-5}	0.9738	0.9887	2.130×10^{-7}	0.9855	0.9184
	u	0.9947	1.871×10^{-5}	0.9985	4.685×10^{-4}	0.9950	0.9980	2.820×10^{-9}	0.9978	0.9581
	v	0.9933	2.244×10^{-5}	0.9493	1.716×10^{-3}	0.9960	0.9980	2.141×10^{-9}	0.9982	0.9443
mean		−1196.2920	6.166×10^{3}	−1305.9226	3.269×10^{1}	−107.7646	−830.8926	4.298	−322.4940	−420.9029
MSE	ε^t_{xx}	2.655×10^{-7}	4.608×10^{-15}	4.500×10^{-7}	7.618×10^{-10}	1.660×10^{-7}	2.177×10^{-7}	2.073×10^{-17}	2.546×10^{-7}	7.480×10^{-8}
	ε^t_{xy}	4.174×10^{-8}	2.747×10^{-16}	3.847×10^{-8}	6.433×10^{-9}	8.423×10^{-8}	2.062×10^{-8}	6.416×10^{-22}	4.949×10^{-8}	9.025×10^{-8}
	ε^t_{yy}	2.494×10^{-7}	1.581×10^{-16}	5.852×10^{-7}	2.150×10^{-9}	1.500×10^{-7}	1.973×10^{-7}	2.708×10^{-16}	1.793×10^{-7}	3.804×10^{-7}
	ε^p_{xx}	3.694×10^{-7}	5.475×10^{-15}	8.900×10^{-7}	4.957×10^{-9}	1.150×10^{-7}	3.109×10^{-7}	2.740×10^{-19}	3.040×10^{-7}	8.305×10^{-8}
	ε^p_{xy}	1.826×10^{-8}	3.699×10^{-18}	1.893×10^{-8}	5.419×10^{-10}	8.258×10^{-10}	1.224×10^{-8}	6.153×10^{-24}	3.270×10^{-9}	6.756×10^{-9}
	ε^p_{yy}	4.966×10^{-7}	8.827×10^{-15}	2.079×10^{-7}	1.285×10^{-9}	1.712×10^{-7}	4.229×10^{-7}	6.942×10^{-20}	4.130×10^{-7}	1.176×10^{-7}
	ε^p_{zz}	9.798×10^{-9}	1.011×10^{-18}	2.318×10^{-7}	9.893×10^{-9}	5.854×10^{-9}	8.779×10^{-9}	6.376×10^{-22}	8.438×10^{-9}	2.996×10^{-9}
	σ_{xx}	1.599×10^{2}	1.859×10^{4}	3.640×10^{1}	6.144	3.304×10^{3}	2.324×10^{1}	2.649×10^{-2}	1.485×10^{3}	2.504×10^{3}
	σ_{xy}	8.763×10^{1}	1.101×10^{3}	7.371×10^{1}	1.158×10^{1}	5.118×10^{2}	3.585×10^{1}	1.534	3.084×10^{2}	4.852×10^{2}
	σ_{yy}	1.180×10^{4}	6.690×10^{5}	6.809×10^{4}	1.127×10^{3}	1.547×10^{4}	8.593×10^{3}	1.887×10^{3}	9.171×10^{3}	4.165×10^{4}
	σ_{zz}	5.870×10^{2}	1.478×10^{4}	1.069×10^{3}	2.994	8.768×10^{2}	3.739×10^{2}	4.308×10^{2}	4.876×10^{2}	2.737×10^{3}
	u	1.288×10^{-2}	1.115×10^{-4}	3.569×10^{-3}	1.144×10^{-3}	1.217×10^{-2}	4.617×10^{-3}	6.444×10^{-9}	5.283×10^{-3}	1.023×10^{-1}
	v	2.259×10^{-4}	2.552×10^{-8}	1.711×10^{-3}	5.786×10^{-5}	1.339×10^{-4}	6.601×10^{-5}	2.616×10^{-12}	6.196×10^{-5}	1.880×10^{-3}

Table A15. Detailed results for the block compression use case Simulation 1.

		SIMULATION 1								
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.7303	1.285×10^{-1}	−3.1200	1.797	−1.0225	0.4661	1.606×10^{-3}	0.7233	0.0611
	ε^t_{xy}	0.5272	2.271×10^{-1}	−1.8630	3.719×10^{-1}	−0.1080	0.4285	3.330×10^{-4}	0.7319	0.0383
	ε^t_{yy}	0.7301	1.291×10^{-1}	−3.0260	2.761	−1.0094	0.4515	2.165×10^{-4}	0.7236	0.0549
	ε^p_{xx}	0.7267	1.309×10^{-1}	−2.6201	1.502	−0.9605	0.4632	1.165×10^{-3}	0.7213	0.0619
	ε^p_{xy}	0.5208	2.282×10^{-1}	−0.2147	6.512×10^{-1}	−0.0480	0.3885	1.938×10^{-2}	0.7295	0.0348
	ε^p_{yy}	0.7273	1.308×10^{-1}	−0.4142	7.218×10^{-1}	−0.9576	0.4298	1.476×10^{-3}	0.7217	0.0624
	ε^p_{zz}	0.3664	2.504×10^{-1}	0.2890	8.202×10^{-2}	−1.0694	0.2682	1.412×10^{-3}	0.6842	0.0806
	σ_{xx}	0.2825	2.839×10^{-1}	0.8531	3.238×10^{-2}	−0.5722	0.7233	2.191×10^{-4}	0.8244	0.1672
	σ_{xy}	0.2157	4.538×10^{-1}	0.8347	8.739×10^{-3}	0.0408	0.5910	4.449×10^{-4}	0.7997	0.1585
	σ_{yy}	0.2810	2.974×10^{-1}	0.7870	2.325×10^{-2}	0.7210	0.6808	3.475×10^{-4}	0.7726	−0.3058
	σ_{zz}	0.3929	3.008×10^{-1}	0.8048	3.480×10^{-3}	0.6143	0.6356	2.169×10^{-4}	0.8023	−0.1644
	u	0.8266	7.245×10^{-2}	0.9294	4.231×10^{-3}	0.4551	0.9157	4.560×10^{-5}	0.9360	0.5587
	v	0.9031	5.961×10^{-2}	0.9872	2.228×10^{-4}	0.7144	0.9619	1.290×10^{-5}	0.9805	0.5683
mean		0.5562	1.952×10^{-1}	−0.4441	6.066×10^{-1}	−0.2463	0.5695	1.665×10^{-3}	0.7808	0.1059
MSE	ε^t_{xx}	9.226×10^{-4}	4.397×10^{-4}	1.409×10^{-2}	6.146×10^{-3}	6.918×10^{-3}	1.826×10^{-3}	5.492×10^{-6}	9.465×10^{-4}	3.211×10^{-3}
	ε^t_{xy}	2.624×10^{-3}	1.261×10^{-3}	1.589×10^{-2}	2.065×10^{-3}	6.150×10^{-3}	3.172×10^{-3}	1.849×10^{-6}	1.488×10^{-3}	5.338×10^{-3}
	ε^t_{yy}	9.301×10^{-4}	4.449×10^{-4}	1.388×10^{-2}	9.514×10^{-3}	6.925×10^{-3}	1.890×10^{-3}	7.460×10^{-7}	9.526×10^{-4}	3.257×10^{-3}
	ε^p_{xx}	9.391×10^{-4}	4.498×10^{-4}	1.244×10^{-2}	5.160×10^{-3}	6.737×10^{-3}	1.845×10^{-3}	4.004×10^{-6}	9.577×10^{-4}	3.224×10^{-3}
	ε^p_{xy}	2.464×10^{-3}	1.173×10^{-3}	6.245×10^{-3}	3.348×10^{-3}	5.387×10^{-3}	3.143×10^{-3}	9.964×10^{-5}	1.391×10^{-3}	4.962×10^{-3}
	ε^p_{yy}	9.431×10^{-4}	4.524×10^{-4}	4.890×10^{-3}	2.496×10^{-3}	6.769×10^{-3}	1.972×10^{-3}	5.103×10^{-6}	9.625×10^{-4}	3.242×10^{-3}
	ε^p_{zz}	6.827×10^{-8}	2.698×10^{-8}	2.458×10^{-3}	2.836×10^{-4}	2.230×10^{-7}	7.885×10^{-8}	1.521×10^{-10}	3.403×10^{-8}	9.906×10^{-8}
	σ_{xx}	1.703×10^{4}	6.737×10^{3}	3.486×10^{3}	7.684×10^{2}	3.732×10^{4}	6.568×10^{3}	5.200	4.168×10^{3}	1.977×10^{4}
	σ_{xy}	1.182×10^{4}	6.839×10^{3}	2.491×10^{3}	1.317×10^{2}	1.446×10^{4}	6.164×10^{3}	6.705	3.019×10^{3}	1.268×10^{4}
	σ_{yy}	9.018×10^{4}	3.730×10^{4}	2.671×10^{4}	2.916×10^{3}	3.499×10^{4}	4.003×10^{4}	4.358×10^{1}	2.852×10^{4}	1.638×10^{5}
	σ_{zz}	1.872×10^{4}	9.276×10^{3}	6.021×10^{3}	1.073×10^{2}	1.189×10^{4}	1.124×10^{4}	6.688	6.098×10^{3}	3.591×10^{4}
	u	3.141×10^{-1}	1.312×10^{-1}	1.278×10^{-1}	7.662×10^{-3}	9.869×10^{-1}	1.526×10^{-1}	8.258×10^{-5}	1.159×10^{-1}	7.993×10^{-1}
	v	4.938×10^{-1}	3.039×10^{-1}	6.544×10^{-2}	1.136×10^{-3}	1.456	1.943×10^{-1}	6.575×10^{-5}	9.955×10^{-2}	2.201

Table A16. Detailed results for the block compression use case Simulation 2.

					SIMULATION 2					
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.7096	2.194×10^{-1}	−1.8843	1.366×10^{-1}	−0.2305	0.5318	2.513×10^{-3}	0.7129	0.0908
	ε^t_{xy}	0.5973	3.255×10^{-1}	−1.0111	5.049×10^{-1}	−0.1731	0.2782	2.402×10^{-3}	0.6317	0.0484
	ε^t_{yy}	0.7065	2.224×10^{-1}	0.4236	2.569×10^{-1}	−0.2238	0.5362	1.582×10^{-3}	0.7144	0.0893
	ε^p_{xx}	0.7009	2.318×10^{-1}	−1.1047	4.716×10^{-1}	−0.1968	0.5474	7.692×10^{-4}	0.7105	0.0898
	ε^p_{xy}	0.6101	3.059×10^{-1}	0.6086	2.773×10^{-2}	−0.0903	0.2472	3.168×10^{-3}	0.6283	0.0483
	ε^p_{yy}	0.7005	2.322×10^{-1}	−0.0181	3.360×10^{-2}	−0.2002	0.5343	8.637×10^{-4}	0.7113	0.0899
	ε^p_{zz}	−0.5742	9.762×10^{-1}	0.6019	7.576×10^{-2}	−1.0839	0.3289	7.350×10^{-4}	0.7009	0.0660
	σ_{xx}	0.3274	3.777×10^{-1}	0.8818	2.484×10^{-3}	−1.6968	0.6480	3.008×10^{-4}	0.8516	0.3220
	σ_{xy}	−0.4305	9.030×10^{-1}	0.5625	6.341×10^{-3}	−0.5689	0.0328	2.754×10^{-4}	0.5789	0.2049
	σ_{yy}	0.1829	4.317×10^{-1}	0.7002	3.199×10^{-3}	0.5536	0.6007	2.593×10^{-5}	0.7000	−0.1031
	σ_{zz}	−0.3596	8.129×10^{-1}	0.7370	1.084×10^{-2}	0.4829	0.5160	1.780×10^{-4}	0.7096	−0.1446
	u	0.8932	1.158×10^{-1}	0.9263	8.374×10^{-3}	0.3561	0.9418	7.221×10^{-5}	0.9579	0.4798
	v	0.8345	1.753×10^{-1}	0.9813	4.156×10^{-3}	0.7313	0.9507	2.445×10^{-5}	0.9677	0.5499
mean		0.3768	3.803×10^{-1}	0.1850	5.531×10^{-2}	−0.1800	0.5149	3.320×10^{-4}	0.7366	0.1409
MSE	ε^t_{xx}	1.744×10^{-3}	1.318×10^{-3}	1.732×10^{-2}	8.206×10^{-4}	7.390×10^{-3}	2.812×10^{-3}	1.509×10^{-5}	1.724×10^{-3}	5.461×10^{-3}
	ε^t_{xy}	2.154×10^{-3}	1.741×10^{-3}	1.076×10^{-2}	2.700×10^{-3}	6.274×10^{-3}	3.861×10^{-3}	1.285×10^{-5}	1.970×10^{-3}	5.090×10^{-3}
	ε^t_{yy}	1.772×10^{-3}	1.342×10^{-3}	3.479×10^{-3}	1.550×10^{-3}	7.387×10^{-3}	2.799×10^{-3}	9.547×10^{-6}	1.724×10^{-3}	5.496×10^{-3}
	ε^p_{xx}	1.811×10^{-3}	1.404×10^{-3}	1.275×10^{-2}	2.856×10^{-3}	7.248×10^{-3}	2.741×10^{-3}	4.658×10^{-6}	1.753×10^{-3}	5.512×10^{-3}
	ε^p_{xy}	1.859×10^{-3}	1.459×10^{-3}	1.866×10^{-3}	1.322×10^{-4}	5.199×10^{-3}	3.590×10^{-3}	1.511×10^{-5}	1.772×10^{-3}	4.538×10^{-3}
	ε^p_{yy}	1.824×10^{-3}	1.414×10^{-3}	6.200×10^{-3}	2.046×10^{-4}	7.309×10^{-3}	2.836×10^{-3}	5.260×10^{-6}	1.758×10^{-3}	5.542×10^{-3}
	ε^p_{zz}	1.408×10^{-7}	8.729×10^{-8}	2.424×10^{-3}	4.614×10^{-4}	1.863×10^{-7}	6.001×10^{-8}	6.573×10^{-11}	2.675×10^{-8}	8.352×10^{-8}
	σ_{xx}	1.019×10^{4}	5.721×10^{3}	1.791×10^{3}	3.762×10^{1}	4.085×10^{4}	5.332×10^{3}	4.557	2.249×10^{3}	1.027×10^{4}
	σ_{xy}	8.585×10^{3}	5.419×10^{3}	2.626×10^{3}	3.805×10^{1}	9.416×10^{3}	5.805×10^{3}	1.653	2.527×10^{3}	4.772×10^{3}
	σ_{yy}	8.018×10^{4}	4.236×10^{4}	2.942×10^{4}	3.139×10^{2}	4.381×10^{4}	3.919×10^{4}	2.545	2.944×10^{4}	1.083×10^{5}
	σ_{zz}	2.661×10^{4}	1.591×10^{4}	5.149×10^{3}	2.122×10^{2}	1.012×10^{4}	9.474×10^{3}	3.484	5.684×10^{3}	2.240×10^{4}
	u	4.301×10^{-1}	4.665×10^{-1}	2.971×10^{-1}	3.374×10^{-2}	2.594	2.343×10^{-1}	2.909×10^{-4}	1.697×10^{-1}	2.096
	v	8.877×10^{-1}	9.405×10^{-1}	1.001×10^{-1}	2.229×10^{-2}	1.442	2.647×10^{-1}	1.312×10^{-4}	1.735×10^{-1}	2.414

Table A17. Detailed results for the block compression use case Simulation 7.

					SIMULATION 7					
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.9991	2.601×10^{-4}	0.9881	6.498×10^{-3}	0.5368	0.9664	2.355×10^{-2}	0.9688	0.3910
	ε^t_{xy}	0.9987	6.959×10^{-4}	0.9623	2.560×10^{-2}	0.2573	0.9512	3.642×10^{-2}	0.9508	0.2373
	ε^t_{yy}	0.9991	2.628×10^{-4}	0.9696	1.661×10^{-2}	0.5397	0.9663	2.395×10^{-2}	0.9691	0.3940
	ε^p_{xx}	0.9991	2.756×10^{-4}	0.8726	4.550×10^{-3}	0.5257	0.9655	2.371×10^{-2}	0.9688	0.3834
	ε^p_{xy}	0.9986	7.165×10^{-4}	0.9616	3.809×10^{-3}	0.2496	0.9577	2.802×10^{-2}	0.9484	0.2319
	ε^p_{yy}	0.9991	2.774×10^{-4}	0.8969	4.043×10^{-3}	0.5281	0.9635	2.655×10^{-2}	0.9690	0.3846
	ε^p_{zz}	0.9968	4.519×10^{-5}	0.7795	9.273×10^{-3}	0.6862	0.9629	2.088×10^{-2}	0.9789	0.4545
	σ_{xx}	0.9965	6.357×10^{-4}	0.9883	6.623×10^{-4}	0.7980	0.9743	1.251×10^{-2}	0.9868	0.5950
	σ_{xy}	0.9962	9.610×10^{-4}	0.9775	1.717×10^{-3}	0.6131	0.9410	3.089×10^{-2}	0.9772	0.5081
	σ_{yy}	0.9915	3.103×10^{-3}	0.8713	1.406×10^{-3}	0.8900	0.9891	4.567×10^{-3}	0.9945	0.7272
	σ_{zz}	0.9943	2.068×10^{-3}	0.9874	3.501×10^{-5}	0.8345	0.9800	1.025×10^{-2}	0.9906	0.6242
	u	0.9996	3.315×10^{-5}	0.9851	1.746×10^{-3}	0.9386	0.9973	1.892×10^{-3}	0.9967	0.9069
	v	0.9997	2.460×10^{-5}	0.9923	2.524×10^{-4}	0.9415	0.9976	1.783×10^{-3}	0.9972	0.9224
mean		0.9976	8.258×10^{-5}	0.9410	4.310×10^{-3}	0.6415	0.9702	1.884×10^{-2}	0.9767	0.5200
MSE	ε^t_{xx}	3.776×10^{-6}	1.124×10^{-6}	5.131×10^{-5}	2.808×10^{-5}	2.001×10^{-3}	1.454×10^{-4}	1.018×10^{-4}	1.349×10^{-4}	2.631×10^{-3}
	ε^t_{xy}	7.595×10^{-6}	4.066×10^{-6}	2.200×10^{-4}	1.496×10^{-4}	4.339×10^{-3}	2.854×10^{-4}	2.128×10^{-4}	2.873×10^{-4}	4.456×10^{-3}
	ε^t_{yy}	3.777×10^{-6}	1.144×10^{-6}	1.324×10^{-4}	7.231×10^{-5}	2.004×10^{-3}	1.467×10^{-4}	1.043×10^{-4}	1.347×10^{-4}	2.638×10^{-3}
	ε^p_{xx}	3.881×10^{-6}	1.196×10^{-6}	5.530×10^{-4}	1.975×10^{-5}	2.059×10^{-3}	1.496×10^{-4}	1.029×10^{-4}	1.356×10^{-4}	2.676×10^{-3}
	ε^p_{xy}	7.212×10^{-6}	3.779×10^{-6}	2.027×10^{-4}	2.009×10^{-5}	3.957×10^{-3}	2.229×10^{-4}	1.478×10^{-4}	2.722×10^{-4}	4.051×10^{-3}
	ε^p_{yy}	3.894×10^{-6}	1.213×10^{-6}	4.511×10^{-4}	1.769×10^{-5}	2.065×10^{-3}	1.598×10^{-4}	1.162×10^{-4}	1.357×10^{-4}	2.692×10^{-3}
	ε^p_{zz}	5.595×10^{-10}	7.808×10^{-12}	9.648×10^{-4}	4.057×10^{-5}	5.421×10^{-8}	6.411×10^{-9}	3.608×10^{-9}	3.652×10^{-9}	9.424×10^{-8}
	σ_{xx}	1.240×10^{2}	2.269×10^{1}	4.180×10^{2}	2.364×10^{1}	7.209×10^{3}	9.186×10^{2}	4.463×10^{2}	4.708×10^{2}	1.445×10^{4}
	σ_{xy}	6.742×10^{1}	1.684×10^{1}	3.950×10^{2}	3.009×10^{1}	6.781×10^{3}	1.034×10^{3}	5.413×10^{2}	3.991×10^{2}	8.620×10^{3}
	σ_{yy}	1.877×10^{3}	6.894×10^{2}	2.858×10^{4}	3.124×10^{2}	2.443×10^{4}	2.421×10^{3}	1.015×10^{3}	1.226×10^{3}	6.060×10^{4}
	σ_{zz}	2.603×10^{2}	9.521×10^{1}	5.801×10^{2}	1.612	7.619×10^{3}	9.203×10^{2}	4.717×10^{2}	4.317×10^{2}	1.730×10^{4}
	u	1.251×10^{-3}	9.453×10^{-5}	4.255×10^{-2}	4.980×10^{-3}	1.752×10^{-1}	7.838×10^{-3}	5.395×10^{-3}	9.348×10^{-3}	2.655×10^{-1}
	v	1.509×10^{-3}	1.275×10^{-4}	4.010×10^{-2}	1.308×10^{-3}	3.030×10^{-1}	1.242×10^{-2}	9.241×10^{-3}	1.433×10^{-2}	4.024×10^{-1}

Table A18. Detailed results for the block compression use case Simulation 12

		SIMULATION 12								
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.6584	3.166×10^{-1}	−3.4145	2.071	−1.1624	0.4656	2.148×10^{-4}	0.7187	0.0506
	ε^t_{xy}	0.5324	3.411×10^{-1}	−1.7259	5.343×10^{-1}	−0.1194	0.4278	4.223×10^{-4}	0.7196	0.0415
	ε^t_{yy}	0.6563	3.215×10^{-1}	−2.9570	2.652	−1.1091	0.4534	8.915×10^{-4}	0.7196	0.0503
	ε^p_{xx}	0.6578	3.175×10^{-1}	−2.7614	1.762	−1.1139	0.4618	1.278×10^{-3}	0.7158	0.0478
	ε^p_{xy}	0.5168	3.478×10^{-1}	−0.1122	6.743×10^{-1}	−0.0789	0.4009	8.991×10^{-3}	0.7157	0.0401
	ε^p_{yy}	0.6591	3.172×10^{-1}	−0.5161	8.422×10^{-1}	−1.1014	0.4300	2.718×10^{-4}	0.7165	0.0495
	ε^p_{zz}	0.6064	3.871×10^{-1}	0.2723	1.236×10^{-1}	−0.4602	0.2658	7.615×10^{-4}	0.6808	0.0781
	σ_{xx}	0.6182	1.745×10^{-1}	0.8828	4.138×10^{-2}	0.0180	0.7218	3.966×10^{-5}	0.8423	0.1688
	σ_{xy}	0.4799	4.811×10^{-1}	0.8803	1.074×10^{-2}	0.3261	0.5912	8.127×10^{-4}	0.7735	0.1288
	σ_{yy}	0.5583	4.457×10^{-1}	0.8136	9.338×10^{-3}	0.3793	0.6803	2.977×10^{-4}	0.8285	0.1275
	σ_{zz}	0.5919	3.890×10^{-1}	0.9196	1.648×10^{-3}	0.3378	0.6356	1.603×10^{-4}	0.7979	0.0963
	u	0.7277	2.782×10^{-1}	0.9072	6.483×10^{-3}	0.4679	0.9157	7.408×10^{-5}	0.9269	0.5649
	v	0.9310	5.347×10^{-2}	0.9866	1.786×10^{-4}	0.7166	0.9621	2.432×10^{-6}	0.9800	0.5744
mean		0.6303	3.204×10^{-1}	−0.4480	6.652×10^{-1}	−0.2230	0.5702	6.266×10^{-4}	0.7797	0.1553
MSE	ε^t_{xx}	1.092×10^{-3}	1.012×10^{-3}	1.412×10^{-2}	6.623×10^{-3}	6.915×10^{-3}	1.828×10^{-3}	7.349×10^{-7}	8.997×10^{-4}	3.036×10^{-3}
	ε^t_{xy}	2.505×10^{-3}	1.827×10^{-3}	1.460×10^{-2}	2.863×10^{-3}	5.997×10^{-3}	3.176×10^{-3}	2.344×10^{-6}	1.502×10^{-3}	5.135×10^{-3}
	ε^t_{yy}	1.115×10^{-3}	1.043×10^{-3}	1.284×10^{-2}	8.604×10^{-3}	6.842×10^{-3}	1.884×10^{-3}	3.073×10^{-6}	9.098×10^{-4}	3.081×10^{-3}
	ε^p_{xx}	1.098×10^{-3}	1.019×10^{-3}	1.207×10^{-2}	5.655×10^{-3}	6.783×10^{-3}	1.849×10^{-3}	4.391×10^{-6}	9.120×10^{-4}	3.055×10^{-3}
	ε^p_{xy}	2.264×10^{-3}	1.629×10^{-3}	5.210×10^{-3}	3.159×10^{-3}	5.054×10^{-3}	3.080×10^{-3}	4.622×10^{-5}	1.332×10^{-3}	4.497×10^{-3}
	ε^p_{yy}	1.107×10^{-3}	1.030×10^{-3}	4.921×10^{-3}	2.734×10^{-3}	6.821×10^{-3}	1.971×10^{-3}	9.399×10^{-7}	9.202×10^{-4}	3.085×10^{-3}
	ε^p_{zz}	1.229×10^{-7}	1.208×10^{-7}	2.362×10^{-3}	4.012×10^{-4}	4.558×10^{-7}	7.910×10^{-8}	8.205×10^{-11}	9.964×10^{-8}	2.878×10^{-7}
	σ_{xx}	2.769×10^{4}	1.265×10^{4}	8.502×10^{3}	3.001×10^{3}	7.121×10^{4}	6.602×10^{3}	9.414×10^{-1}	1.144×10^{4}	6.028×10^{4}
	σ_{xy}	2.415×10^{4}	2.233×10^{4}	5.559×10^{3}	4.987×10^{2}	3.128×10^{4}	6.161×10^{3}	1.225×10^{1}	1.052×10^{4}	4.045×10^{4}
	σ_{yy}	1.760×10^{5}	1.776×10^{5}	7.428×10^{4}	3.722×10^{3}	2.474×10^{5}	4.010×10^{4}	3.734×10^{1}	6.835×10^{4}	3.478×10^{5}
	σ_{zz}	3.808×10^{4}	3.630×10^{4}	7.501×10^{3}	1.538×10^{2}	6.179×10^{4}	1.124×10^{4}	4.943	1.886×10^{4}	8.433×10^{4}
	u	4.605×10^{-1}	4.704×10^{-1}	1.569×10^{-1}	1.096×10^{-2}	8.999×10^{-1}	1.527×10^{-1}	1.342×10^{-4}	1.235×10^{-1}	7.358×10^{-1}
	v	3.434×10^{-1}	2.660×10^{-1}	6.647×10^{-2}	8.887×10^{-4}	1.410	1.934×10^{-1}	1.240×10^{-5}	9.955×10^{-2}	2.117

Table A19. Detailed results for the block compression use case Simulation 13.

		SIMULATION 13								
		MLP		**PINN**		**SVR**	**GBDTR**		**KNNR**	**GPR**
		mean	**std**	**mean**	**std**		**mean**	**std**		
R2	ε^t_{xx}	0.7511	2.314×10^{-1}	−4.3817	2.685	−0.2933	0.5204	9.591×10^{-4}	0.7164	0.0857
	ε^t_{xy}	0.5801	3.853×10^{-1}	−0.9114	1.065×10^{-1}	−0.1797	0.2623	2.829×10^{-3}	0.6314	0.0485
	ε^t_{yy}	0.7517	2.319×10^{-1}	0.1125	9.052×10^{-2}	−0.2646	0.5280	1.380×10^{-3}	0.7186	0.0873
	ε^p_{xx}	0.7547	2.273×10^{-1}	−2.1130	9.596×10^{-1}	−0.2740	0.5357	4.099×10^{-4}	0.7131	0.0826
	ε^p_{xy}	0.5616	4.021×10^{-1}	0.5356	2.972×10^{-2}	−0.1116	0.2250	7.643×10^{-3}	0.6277	0.0480
	ε^p_{yy}	0.7556	2.269×10^{-1}	−0.4174	3.678×10^{-1}	−0.2741	0.5217	8.630×10^{-4}	0.7142	0.0830
	ε^p_{zz}	0.6959	2.525×10^{-1}	0.4548	2.042×10^{-1}	−0.4013	0.4990	7.038×10^{-5}	0.7267	0.0912
	σ_{xx}	0.4251	5.887×10^{-1}	0.9077	3.046×10^{-3}	−0.4988	0.7847	3.553×10^{-4}	0.8620	0.2942
	σ_{xy}	−0.0088	8.836×10^{-1}	0.8314	3.051×10^{-3}	0.1802	0.4255	6.447×10^{-4}	0.6805	0.1513
	σ_{yy}	0.6042	4.473×10^{-1}	0.6678	8.365×10^{-4}	−0.0250	0.7608	5.706×10^{-6}	0.8418	0.1470
	σ_{zz}	0.6356	3.848×10^{-1}	0.9351	1.247×10^{-3}	0.0788	0.6999	1.686×10^{-4}	0.8282	0.1321
	u	0.9627	2.725×10^{-2}	0.9392	4.615×10^{-3}	0.3676	0.9467	1.345×10^{-4}	0.9625	0.4940
	v	0.9478	5.094×10^{-2}	0.9805	3.346×10^{-3}	0.7270	0.9521	1.989×10^{-5}	0.9697	0.5572
mean		0.6475	3.326×10^{-1}	−0.1122	3.265×10^{-1}	−0.0745	0.5894	1.579×10^{-4}	0.7687	0.1771
MSE	ε^t_{xx}	1.412×10^{-3}	1.313×10^{-3}	3.054×10^{-2}	1.523×10^{-2}	7.339×10^{-3}	2.721×10^{-3}	5.442×10^{-6}	1.609×10^{-3}	5.188×10^{-3}
	ε^t_{xy}	2.253×10^{-3}	2.067×10^{-3}	1.025×10^{-2}	5.715×10^{-4}	6.329×10^{-3}	3.958×10^{-3}	1.518×10^{-5}	1.977×10^{-3}	5.105×10^{-3}
	ε^t_{yy}	1.422×10^{-3}	1.328×10^{-3}	5.080×10^{-3}	5.182×10^{-4}	7.239×10^{-3}	2.702×10^{-3}	7.900×10^{-6}	1.611×10^{-3}	5.225×10^{-3}
	ε^p_{xx}	1.389×10^{-3}	1.288×10^{-3}	1.763×10^{-2}	5.436×10^{-3}	7.217×10^{-3}	2.630×10^{-3}	2.322×10^{-6}	1.625×10^{-3}	5.197×10^{-3}
	ε^p_{xy}	2.107×10^{-3}	1.933×10^{-3}	2.232×10^{-3}	1.428×10^{-4}	5.342×10^{-3}	3.724×10^{-3}	3.673×10^{-5}	1.789×10^{-3}	4.575×10^{-3}
	ε^p_{yy}	1.398×10^{-3}	1.299×10^{-3}	8.111×10^{-3}	2.104×10^{-3}	7.291×10^{-3}	2.737×10^{-3}	4.939×10^{-6}	1.635×10^{-3}	5.247×10^{-3}
	ε^p_{zz}	7.358×10^{-8}	6.111×10^{-8}	3.120×10^{-3}	1.168×10^{-3}	3.391×10^{-7}	1.212×10^{-7}	1.703×10^{-11}	6.614×10^{-8}	2.199×10^{-7}
	σ_{xx}	2.766×10^{4}	2.832×10^{4}	4.438×10^{3}	1.466×10^{2}	7.211×10^{4}	1.036×10^{4}	1.709×10^{1}	6.640×10^{3}	3.395×10^{4}
	σ_{xy}	1.954×10^{4}	1.711×10^{4}	3.266×10^{3}	5.908×10^{1}	1.588×10^{4}	1.113×10^{4}	1.249×10^{1}	6.187×10^{3}	1.644×10^{4}
	σ_{yy}	1.214×10^{5}	1.372×10^{5}	1.019×10^{5}	2.565×10^{2}	3.143×10^{5}	7.333×10^{4}	1.750	4.851×10^{4}	2.615×10^{5}
	σ_{zz}	2.089×10^{4}	2.205×10^{4}	3.720×10^{3}	7.147×10^{1}	5.280×10^{4}	1.720×10^{4}	9.664	9.844×10^{3}	4.974×10^{4}
	u	1.395×10^{-1}	1.019×10^{-1}	2.273×10^{-1}	1.726×10^{-2}	2.365	1.992×10^{-1}	5.028×10^{-4}	1.401×10^{-1}	1.892
	v	2.707×10^{-1}	2.644×10^{-1}	1.011×10^{-1}	1.737×10^{-2}	1.417	2.487×10^{-1}	1.032×10^{-4}	1.573×10^{-1}	2.299

References

1. Reddy, J.N. *Introduction to the Finite Element Method*; Mechanical Engineering; McGraw Hill Education: New York, NY, USA, 2019.
2. Yang, X.-S.; Koziel, S.; Leifsson, L. Computational Optimization, Modelling and Simulation: Recent Trends and Challenges. *Procedia Comput. Sci.* **2013**, *18*, 855–860. [CrossRef]
3. Roberts, S.M.; Kusiak, J.; Liu, Y.L.; Forcellese, A.; Withers, P.J. Prediction of damage evolution in forged aluminium metal matrix composites using a neural network approach. *J. Mater. Process. Technol.* **1998**, *80–81*, 507–512. [CrossRef]
4. Hans Raj, K.; Sharma, R.S.; Srivastava, S.; Patvardhan, C. Modeling of manufacturing processes with ANNs for intelligent manufacturing. *Int. J. Mach. Tools Manuf.* **2000**, *40*, 851–868. [CrossRef]
5. García-Crespo, A.; Ruiz-Mezcua, B.; Fernández-Fdz, D.; Zaera, R. Prediction of the response under impact of steel armours using a multilayer perceptron. *Neural Comput. Appl.* **2007**, *16*, 147–154. [CrossRef]
6. Nourbakhsh, M.; Irizarry, J.; Haymaker, J. Generalizable surrogate model features to approximate stress in 3D trusses. *Eng. Appl. Artif. Intell.* **2018**, *71*, 15–27. [CrossRef]
7. Chan, W.L.; Fu, M.W.; Lu, J. An integrated FEM and ANN methodology for metal-formed product design. *Eng. Appl. Artif. Intell.* **2008**, *21*, 1170–1181. [CrossRef]
8. D'Addona, D.M.; Antonelli, D. Neural Network Multiobjective Optimization of Hot Forging. *Procedia CIRP* **2018**, *67*, 498–503. [CrossRef]
9. Gudur, P.P.; Dixit, U.S. A neural network-assisted finite element analysis of cold flat rolling. *Eng. Appl. Artif. Intell.* **2008**, *21*, 43–52. [CrossRef]
10. Pellicer-Valero, O.J.; Rupérez, M.J.; Martínez-Sanchis, S.; Martín-Guerrero, J.D. Real-time biomechanical modeling of the liver using Machine Learning models trained on Finite Element Method simulations. *Expert Syst. Appl.* **2020**, *143*, 113083. [CrossRef]
11. Abueidda, D.W.; Almasri, M.; Ammourah, R.; Ravaioli, U.; Jasiuk, I.M.; Sobh, N.A. Prediction and optimization of mechanical properties of composites using convolutional neural networks. *Compos. Struct.* **2019**, *227*, 111264. [CrossRef]
12. Pfaff, T.; Fortunato, M.; Sanchez-Gonzalez, A.; Battaglia, P.W. Learning Mesh-Based Simulation with Graph Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26 April–1 May 2020.
13. Loghin, A.; Ismonov, S. Augmenting Generic Fatigue Crack Growth Models Using 3D Finite Element Simulations and Gaussian Process Modeling. In Proceedings of the ASME 2019 Pressure Vessels & Piping Conference, San Antonio, TX, USA, 14–19 July 2019; Volume 2. [CrossRef]
14. Ming, W.; Zhang, G.; Li, H.; Guo, J.; Zhang, Z.; Huang, Y.; Chen, Z. A hybrid process model for EDM based on finite-element method and Gaussian process regression. *Int. J. Adv. Manuf. Technol.* **2014**, *74*, 1197–1211. [CrossRef]
15. Pan, F.; Zhu, P.; Zhang, Y. Metamodel-based lightweight design of B-pillar with TWB structure via support vector regression. *Comput. Struct.* **2010**, *88*, 36–44. [CrossRef]
16. Li, H.; Shi, M.; Liu, X.; Shi, Y. Uncertainty optimization of dental implant based on finite element method, global sensitivity analysis and support vector regression. Proceedings of the Institution of Mechanical Engineers. *Part H J. Eng. Med.* **2019**, *233*, 232–243. [CrossRef] [PubMed]
17. Hu, F.; Li, D. Modelling and Simulation of Milling Forces Using an Arbitrary Lagrangian–Eulerian Finite Element Method and Support Vector Regression. *J. Optim. Theory Appl.* **2012**, *153*, 461–484. [CrossRef]
18. Martínez-Martínez, F.; Rupérez-Moreno, M.J.; Martínez-Sober, M.; Solves-Llorens, J.A.; Lorente, D.; Serrano-López, A.J.; Martínez-Sanchis, S.; Monserrat, C.; Martín-Guerrero, J.D. A finite element-based machine learning approach for modeling the mechanical behavior of the breast tissues under compression in real-time. *Comput. Biol. Med.* **2017**, *90*, 116–124. [CrossRef] [PubMed]
19. Zhang, W.; Zhang, R.; Wu, C.; Goh, A.T.C.; Wang, L. Assessment of basal heave stability for braced excavations in anisotropic clay using extreme gradient boosting and random forest regression. *Undergr. Space* **2020**. [CrossRef]
20. Qi, Z.; Zhang, N.; Liu, Y.; Chen, W. Prediction of mechanical properties of carbon fiber based on cross-scale FEM and machine learning. *Compos. Struct.* **2019**, *212*, 199–206. [CrossRef]
21. Haghighat, E.; Juanes, R. SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Comput. Methods Appl. Mech. Eng.* **2021**, *373*, 113552. [CrossRef]
22. Haghighat, E.; Raissi, M.; Moure, A.; Gomez, H.; Juanes, R. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Comput. Methods Appl. Mech. Eng.* **2021**, *379*, 113741. [CrossRef]
23. Shin, Y. On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs. *CiCP* **2020**, *28*, 2042–2074. [CrossRef]
24. Yin, M.; Zheng, X.; Humphrey, J.D.; Em Karniadakis, G. Non-invasive Inference of Thrombus Material Properties with Physics-Informed Neural Networks. *Comput. Methods Appl. Mech. Eng.* **2021**, *375*, 113603. [CrossRef]
25. Arnold, F.; King, R. State–space modeling for control based on physics-informed neural networks. *Eng. Appl. Artif. Intell.* **2021**, *101*, 104195. [CrossRef]
26. Zobeiry, N.; Humfeld, K.D. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Eng. Appl. Artif. Intell.* **2021**, *101*, 104232. [CrossRef]
27. Nascimento, R.G.; Fricke, K.; Viana, F.A.C. A tutorial on solving ordinary differential equations using Python and hybrid physics-informed neural network. *Eng. Appl. Artif. Intell.* **2020**, *96*, 103996. [CrossRef]
28. Jin, X.; Cai, S.; Li, H.; Karniadakis, G.E. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *J. Comput. Phys.* **2021**, *426*, 109951. [CrossRef]

29. Mao, Z.; Jagtap, A.D.; Karniadakis, G.E. Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.* **2020**, *360*, 112789. [CrossRef]
30. Goan, E.; Fookes, C. Bayesian Neural Networks: An Introduction and Survey. In *Case Studies in Applied Bayesian Data Science*; CIRM Jean-Morlet Chair, Fall 2018; Lecture Notes in Mathematics; Mengersen, K.L., Pudlo, P., Robert, C.P., Eds.; Springer: Cham, Switzerland, 2020; Volume 2259, pp. 45–87.
31. Garnelo, M.; Rosenbaum, D.; Maddison, C.; Ramalho, T.; Saxton, D.; Shanahan, M.; Teh, Y.W.; Rezende, D.; Eslami, S.M.A. Conditional Neural Processes. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Dy, J., Krause, A., Eds.; pp. 1704–1713.
32. Wang, S.; Teng, Y.; Perdikaris, P. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv* **2020**, arXiv:2001.04536.
33. Pang, G.; Lu, L.; Karniadakis, G.E. fPINNs: Fractional Physics-Informed Neural Networks. *SIAM J. Sci. Comput.* **2019**, *41*, A2603–A2626. [CrossRef]
34. Yang, Y.; Perdikaris, P. Adversarial uncertainty quantification in physics-informed neural networks. *J. Comput. Phys.* **2019**, *394*, 136–152. [CrossRef]
35. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
36. Ngatchou, P.; Zarei, A.; El-Sharkawi, A. Pareto Multi Objective Optimization. In Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems, Arlington, VA, USA, 6–10 November 2005; IEEE Service Center: Piscataway, NJ, USA, 2005; pp. 84–91. [CrossRef]
37. Pettit, C.L.; Wilson, D.K. A physics-informed neural network for sound propagation in the atmospheric boundary layer. In Proceedings of the 179th Meeting of the Acoustical Society of America, Acoustics Virtually Everywhere, 7–11 December 2020; p. 22002. [CrossRef]
38. Lihua, L. Simulation physics-informed deep neural network by adaptive Adam optimization method to perform a comparative study of the system. *Eng. Comput.* **2021**, 1–20. [CrossRef]
39. McClenny, L.; Braga-Neto, U. Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism. *arXiv* **2020**, arXiv:2009.04544.
40. Jagtap, A.D.; Kawaguchi, K.; Karniadakis, G.E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J. Comput. Phys.* **2020**, *404*, 109136. [CrossRef]
41. 1.11. Ensemble Methods—Scikit-Learn 0.24.1 Documentation (2021.000Z). Available online: https://scikit-learn.org/stable/modules/ensemble.html (accessed on 5 November 2020).
42. Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
43. 1.6. Nearest Neighbors—Scikit-Learn 0.24.1 Documentation (2021.000Z). Available online: https://scikit-learn.org/stable/modules/neighbors.html (accessed on 5 November 2020).
44. Schulz, E.; Speekenbrink, M.; Krause, A. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *J. Math. Psychol.* **2018**, *85*, 1–16. [CrossRef]
45. Rasmussen, C.E.; Williams, C.K.I. *Gaussian Processes for Machine Learning*, 3rd ed.; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 2008.
46. Awad, M.; Khanna, R. Support Vector Regression. In *Efficient Learning Machines. Theories, Concepts, and Applications for Engineers and System Designers*; The Expert's Voice in Machine Learning; Awad, M., Khanna, R., Eds.; Apress Open: New York, NY, USA, 2015; pp. 67–80.
47. Smola, A.J.; Schölkopf, B. A tutorial on support vector regression. *Stat. Comput.* **2004**, *14*, 199–222. [CrossRef]
48. Svozil, D.; Kvasnicka, V.; Pospichal, J. Introduction to multi-layer feed-forward neural networks. *Chemom. Intell. Lab. Syst.* **1997**, *39*, 43–62. [CrossRef]
49. Leshno, M.; Lin, V.Y.; Pinkus, A.; Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Netw.* **1993**, *6*, 861–867. [CrossRef]
50. Lu, Z.; Pu, H.; Wang, F.; Hu, Z.; Wang, L. The Expressive Power of Neural Networks: A View from the Width. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
51. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
52. Gardner, M.W.; Dorling, S.R. Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences. *Atmos. Environ.* **1998**, *32*, 2627–2636. [CrossRef]
53. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
54. Lu, L.; Pestourie, R.; Yao, W.; Wang, Z.; Verdugo, F.; Johnson, S.G. Physics-informed neural networks with hard constraints for inverse design. *arXiv* **2021**, arXiv:2102.04626.