Physics Informed Neural Network for Dynamic Stress Prediction

Hamed Bolandi[1,2*], Gautam Sreekumar[2], Xuyang Li[1,2], Nizar Lajnef[1] and Vishnu Naresh Boddeti[2]

[1*]Civil and Environmental Engineering, Michigan State University, Shaw Lane, East Lansing, 48824, MI, USA.
[2]Computer Science and Engineering, Michigan State University, Shaw Lane, East Lansing, 48824, MI, USA.

*Corresponding author(s). E-mail(s): bolandih@msu.edu;
Contributing authors: sreekum1@msu.edu; lixuyan1@msu.edu; lajnefni@msu.edu; vishnu@msu.edu;

Abstract

Structural failures are often caused by catastrophic events such as earthquakes and winds. As a result, it is crucial to predict dynamic stress distributions during highly disruptive events in real time. Currently available high-fidelity methods, such as Finite Element Models (FEMs), suffer from their inherent high complexity. Therefore, to reduce computational cost while maintaining accuracy, a Physics Informed Neural Network (PINN), PINN-Stress model, is proposed to predict the entire sequence of stress distribution based on Finite Element simulations using a partial differential equation (PDE) solver. Using automatic differentiation, we embed a PDE into a deep neural network's loss function to incorporate information from measurements and PDEs. The PINN-Stress model can predict the sequence of stress distribution in almost real-time and can generalize better than the model without PINN.

Keywords: Physics Informed Neural Network, Stress Prediction, Finite Element Analysis, Partial Differential Equation

1 Introduction

A dynamic analysis is used to determine how a system will respond to general time-dependent loads. Events such as earthquakes and explosions are typical applications for dynamic analysis. These applications should be able to carry out real-time analysis in the aftermath of a disaster or during extreme disruptive events that require immediate corrections to avoid catastrophic failures. Dynamic loading also can cause dramatic and damaging failures, which can be avoided by evaluating the structure during the design phase. Finite Element Analysis (FEA) is a powerful engineering tool used to analyze the behavior of physical systems under different conditions. FEA can be used to predict the behavior of a system by evaluating various parameters such as forces, boundary conditions, stresses, and, displacements. FEA is performed using specialized computer software called a Finite Element Method (FEM) solver. FEA is typically used to conduct dynamic stress analysis of various structures and systems in which it might be hard to determine an analytical solution. However, it is important to note that FEA is a complex process and requires extensive knowledge and experience to use properly and also is computationally prohibitive while being accurate. The current workflow for FEA applications consists of (i) modeling the geometry and its components, (ii) specifying the material properties, boundary conditions, meshing, and loading, (iii) dynamic analysis, which may be time-consuming based on the complexity of the model. The complexity of this workflow and its time requirements make it impractical for real-time applications.

The recently introduced models [1, 2] were designed to predict static stress distributions using deep neural network (DNN)-based methods in both intact and damaged structural components. The primary limitations of the above data-driven models are the incapability to produce physically consistent results and the lack of generalizability to out-of-distribution scenarios. The concept of physics-informed learning was introduced recently [3–5] to address the computational cost of FEA and lack of generalizability to out-of-distribution scenarios. There is special interest in Physics-informed Neural Networks (PINNs), which directly incorporate partial differential equations (PDEs) into the training loss function. However, their applications have primarily been limited to non-engineering toy simulations. Working with engineering problems such as those in structural engineering will require these models to learn several factors of variation in addition to the physical equations themselves, such as geometry. To overcome these issues, we propose a novel model for dynamic stress prediction in the specific domain of 2D steel plates which is real-time and generalizable and can therefore be used for stress prediction in seismic and explosions design.

We augment PINN with a novel neural architecture for predicting dynamic stress distribution to achieve fast dynamic analysis and address deficiencies of data-driven models. We model the stress distribution in gusset plates under dynamic loading to demonstrate its utility. Gusset plates are one of the most

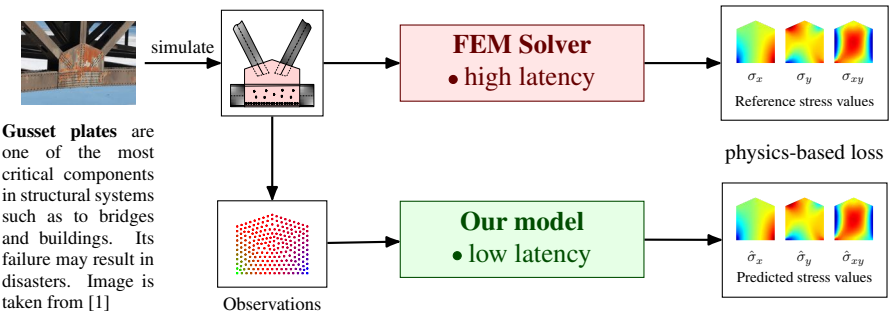


Fig. 1: **Overview:** Unlike FEM, PINN-Stress is computationally efficient, facilitates real-time analysis and is generalizable. PINN-Stress use a governing equation behind the equation of motion as a soft constraint in the loss function to enforce the loss to minimize. The points with different colors in observations correspond to the same nodes in the gusset plate. Gusset plate image is taken from [6]

critical components in structural systems such as bridges and buildings. Since gusset plates are designed for lateral loads such as earthquakes, wind, and explosions, real-time dynamic models like ours can help avoid catastrophic failures. In practice, the outputted stress maps from our models can be used by downstream applications to detect anomalies such as plate cracks. In other words, it can be a precursor to existing vision-based systems.

An overview of our approach is shown in Fig. 1. To summarize our contributions, we introduce NeuroStress and PINN-Stress, two novel deep learning models to learn dynamic stress distribution for complex geometries, boundary conditions, and various load sequences. The loss function in NeuroStress uses MAE loss as defined in Eq. 10 in section 5.2 for training. PINN-Stress uses the physics-informed loss function described in Section 3.1. Our models require input from sensors placed on the plates for real-life use. But since it is difficult to obtain such data for research purposes, we generate challenging synthetic data emulating dynamic stress prediction. Through extensive experiments on simulated data, we show that:

1. NeuroStress and PINN-Stress can predict dynamic stress distribution with complex geometries, boundary conditions, and various load sequences faster than traditional FEA solvers. Previous works only predict static stress distribution;
2. NeuroStress and PINN-Stress can learn the temporal information in the data to make accurate predictions;
3. NeuroStress and PINN-Stress can predict von Mises stress distribution using the von Mises equation. von Mises stress distribution is a primary diagnostic tool to predict the failure of a structure;
4. Introducing a differentiable grid as a surrogate grid to calculate gradients of stress output along horizontal and vertical directions.

5. To the best of our knowledge, PINN-Stress is the first model that learns governing equations behind that of motion in structures. We attribute the generalization abilities of our architecture on unseen load sequences and geometries to its loss function.

2 Related Works

Over the past few years, there has been a revolution in data-driven applications in various engineering fields, including fluid dynamics [7, 8], molecular dynamics simulation [9, 10] and material properties prediction [11–14]. Recent studies have shown that convolutional neural networks (CNN) and Long Term Short Memories (LSTM) can be used to build metamodels for predicting time history responses. Modares et al. [15] studied composite materials to identify the presence and type of structural damage using CNNs. Nie et al. [16] developed a CNN-based method to predict the low-resolution stress field in a 2D linear cantilever beam. Jiang et al. [17] developed a conditional generative adversarial network for predicting low-resolution static von Mises stress distribution in solid structures. Zhang et al. [18] used LSTM to model nonlinear seismic responses of structures with large plastic deformations. Do et al. [19] proposed a method for forecasting crack propagation in risk assessment of engineering structures based on LSTM and Multi-Layer Perceptron (MLP). Presas et al [20] proposed a neural network to estimate the magnitude of static and dynamic stresses based on the measurements of stationary sensors in turbines. Raissi et al [3, 21] used Gaussian process regression to construct representations of linear operator functionals. Their model can accurately infer the solution and provide uncertainty estimates for different physical problems; this was then extended in [4, 22]. Raissi et al. [23] proposed a physics-informed neural network that can solve supervised learning tasks while respecting any given law of physics described by general nonlinear partial differential equations. For solving nonlinear PDEs, such as Schrödinger, Burgers, and Allen–Cahn equations, Raissi et al [24] introduced and illustrated the PINN approach. Vahab et al [25] developed Physics-Informed Neural Networks based on Airy stress functions and Fourier series to find optimal solutions to a few reference biharmonic problems of elasticity and elastic plate theory. Yan et al [26] proposed an approach to solving linear elasticity problems in composite plates and tubes using Physics Informed Neural Network. Chen et al [27] proposed a PINN for fatigue life prediction with a sparse amount of experimental data combined with physical models describing the fatigue behavior of materials. Bai et al [28] proposed an advanced PINN method based on the modified loss function for computational 2D and 3D solid mechanics. Jeong et al [29] introduced a Physics-Informed Neural Network-based Topology Optimization (PINNTO) framework which is a combination of Topology Optimization and Physics-Informed PINNs. PINNTO uses an energy-based PINN to replace FEA in the conventional structural topology optimization and numerically determine the deformation states. Zhang et al [30] presented a PINN method for

identifying unknown geometric and material parameters. They parameterize the geometry of the material using a mesh-free method and a differentiable and trainable technique that can identify multiple structural features. Fallah et al [31] proposed a PINN model for bending and free vibration analysis of three-dimensional functionally graded porous beams. Bazmara et al [32] built a PINN framework using the Euler-Bernoulli beam theory and Hamilton principle to predict the nonlinear bending of the beam system. Zaho et al [33] presented a PINN model for temperature field predicting heat source layout. Xu et al [34] introduced a PINN model for predicting external loads of diverse engineering structures based on limited displacement monitoring points. Zheng et al [35] reconstructed the solution of the displacement field after damage to predict crack propagation using PINN. Yao et al. [36] proposed a physics-guided learning algorithm for predicting the mechanical response of materials and structures. Das et al. [37] proposed a data-driven physics-informed method for prognosis and applied it to predict cracking in a mortar cube specimen. Wang et al. [38] proposed a hybrid DL model that unifies representation learning and turbulence simulation techniques using physics-informed learning. Goswami et al. [39] proposed a physics-informed variational formulation of DeepONet for brittle fracture analysis. Haghighat et al. [40] presented physics-informed neural networks to inversion and surrogate modeling in solid mechanics. Jin et al. [41] investigated the ability of PINNs to directly simulate incompressible flows, ranging from laminar to turbulent flows to turbulent channel flows. Li et al. [42] used the Fourier transform to develop a Fourier neural operator to model turbulent flows.

3 Background

3.1 Stress equilibrium equation

To ensure that any component of an object is in equilibrium, the balance of forces and moments acting on that component should be enforced. Stress components acting on the face of the element can be written as equations of equilibrium. The stress equilibrium equation can be written as a variation in each stress term within the body since stress changes from point to point. Considering a two-dimensional case in which stress acts in the horizontal and vertical directions gives the following set of equations of motion:

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + b_x - \rho a_x = 0 \tag{1}$$

$$\frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{xy}}{\partial x} + b_y - \rho a_y = 0 \tag{2}$$

where σ_{xx}, σ_{yy} and σ_{xy} denote normal stress in horizontal and vertical directions, and shear stress respectively. b_x and b_y represent body force in horizontal and vertical directions. a_x and a_y represent an acceleration in the horizontal and vertical directions and ρ denotes the density of the material.

3.2 Stress Calculation

The steps for linear finite element analysis' stress calculation, which is part of phase (iii) of FEA's workflow elaborated in the introduction section, are as follows:

$$KQ = F \tag{3}$$

where K denotes a global stiffness matrix, F is the load vector applied at each node, and Q denotes the displacement. A stiffness matrix K consists of elemental stiffness matrices K_e:

$$K_e = A_e B^T D B \tag{4}$$

where B represents strain-displacement matrix; D represents stress-strain matrix; and A_e represents area of element. Mesh geometry and material properties determine B and D. This will be followed by adding the local stiffness matrix k_e to the global stiffness matrix. The displacement boundary conditions are encoded using the corresponding rows and columns in the global stiffness matrix K. Solving Q can be achieved using direct factorization or iterative methods.

As a result of calculating the global displacement using equation 3, we can calculate the nodal displacements q then we can calculate the stress tensors of each element as follows:

$$\sigma = DBq \tag{5}$$

von Mises stress is a way of measuring whether a structure has begun to yield at any point. To compare experimentally observed yield points with calculated stresses, von Mises stress can be used mathematically as a scalar quantity. We also predict von Mises stress since the engineering community relies heavily on it. von Mises stress can be calculated from the predicted σ_{xx}, σ_{yy}, and σ_{xy} through the von Mises stress equation.

$$\sigma_{vm} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 - \sigma_{xx}\sigma_{yy} + 3\sigma_{yy}^2} \tag{6}$$

4 Method

We introduce a novel architecture in this paper and augment it with a physics-based loss function for gains in generalization.

4.1 Architecture

Firstly, we use a 2-layered MLP to encode the input to a larger dimensional space. Then we introduce our spatiotemporal multiplexing (STM) module to encode the spatial and temporal information alternatively. We treat both the temporal and the spatial dimensions as sequences, which may be modeled using an appropriate deep neural architecture such as RNN, LSTM [43] or

self-attention [44]. A conventional RNN is vulnerable to vanishing gradient and exploding gradient issues [45] when trained with gradient-based learning and back-propagation through time (BPTT) [46]. LSTM has been developed by Hochreiter et al [43] to address this problem. The LSTM architecture is an RNN architecture that is capable of retaining both short- and long-term dependencies. Compared with traditional RNNs and LSTM, Transformer architecture can train faster due to parallelization, perform better on certain tasks, handle long sequences more efficiently, and scale better to larger datasets and models. Furthermore, transformer training is generally more stable than LSTM training [47]. Based on the above arguments LSTMs have demonstrated better performance than RNNs, but have performed worse compared to self-attention. However, self-attention requires plenty of data, which cannot be satisfied in our problem statement. Hence, as a middle ground, we use LSTMs to model both temporal and spatial information.

Spatiotemporal multiplexing (STM): A single instance of our STM module consists of two LSTM layers - one for temporal sequence modeling and another for spatial sequence modeling. The input feature to an STM module is of shape $B \times N \times T \times d$ where B, N, T, d are batch size, number of spatial nodes, number of time frames and feature dimension, respectively. We reshape this tensor into $BN \times T \times d$ and feed it as input to the first LSTM. Here, T forms the index for sequence. The output tensor from this LSTM is reshaped to $BT \times N \times d$ before feeding it into the second LSTM for spatial sequence modeling. The network layers are summarized in Table 1. We would like to point out that the idea of multiplexing is not novel in deep learning literature [48, 49]. Our contribution is that we used multiplexing in physics-informed learning and show its utility in dynamic prediction. Our whole architecture consists of three STM modules, totaling six LSTM layers. The architecture is schematically shown in Fig. 2

Table 1: Network layers

Item	Number of layers	Input dimension	Output dimension	Activation
MLP Encoder	2	5×64	B×N×T×d	ReLU
Spatial	3	B∗N×T×d	B∗T×N×d	
Temporal	3	B∗T×N×d	B∗N×T×d	
MLP Decoder	1	64×3	B×N×T×d	ReLU

* B: batch size, N: number of nodes, T: time frames, d: feature dimension

4.2 Physics Loss Function

In order to force our model to learn the physical constraints, we minimize the violation of the physical equations shown in Eq. 1 and 2. We also minimize the boundary condition violation to fully enforce the underlying PDE. Specifically, our loss function is a weighted sum of three loss terms:

$$\mathcal{L} = w_{\text{data}}\mathcal{L}_{\text{data}} + w_{\text{PDE}}\mathcal{L}_{\text{PDE}} + w_{\text{bc}}\mathcal{L}_{\text{bc}} \tag{7}$$

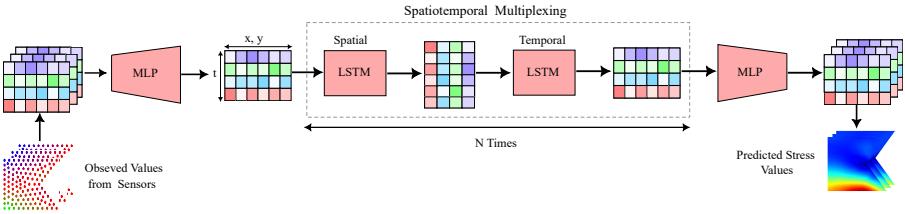


Fig. 2: **Model architecture:** We introduce the novel spatiotemporal multiplexing (STM) to physics-informed learning in order to learn both spatial and temporal information in the data. Our architecture is lightweight and hence gives real-time performance.

where $\mathcal{L}_{\text{data}}$ measures the mean absolute error (MAE) between true and predicted labels. $\mathcal{L}_{\text{PDE}}$ measures the violations of the physical equations defined in Eq. 1 and 2 by calculating the mean absolute error between the Left Hand Side (LHS) and the Right Hand Side (RHS). $\mathcal{L}_{\text{bc}}$ corresponds to boundary condition constraints. w_{data}, w_{PDE}, and w_{bc} are the weights used to balance the interplay between the three loss terms. $\mathcal{L}_{\text{bc}}$ consists of the initial and boundary conditions at each time step as below:

$$\sigma(x, y, t = 0) = 0 \tag{8}$$

$$\sigma(x, y, (t_0 ... t_n)) = \sigma \tag{9}$$

Equations 8 and 9 should be satisfied for σ_{xx}, σ_{yy} and σ_{xy}. x and y are coordinates of meshes in each sample, and t is the time at time steps.

4.3 Differentiable grid from mesh

Our physics-based loss function requires us to estimate the gradients of stress output along x and y directions. But since our output is in the form of a triangular mesh, gradient computation is not easy. Instead, we propose a novel method for calculating gradients on a surrogate grid created using kernel density estimation (KDE). Specifically, we calculate the stress value at a grid vertex by adding contributions from every mesh node, weighted by a Gaussian filter centered at this vertex and having a specific variance. By tuning the variance of this filter, we can achieve a robust, accurate reconstruction of the mesh along with a mask showing extrapolated regions. The original mesh, the grid reconstructed from it, and the corresponding mask are shown in Fig. 3. To compare the accuracy of the surrogate grid, we compare it against the reconstruction obtained through `tricontourf` function in Matplotlib package in Python. As can be observed in Fig. 3c, the grid is accurate within the mesh region. Now, we can estimate the gradients for the stress outputs from these grids.

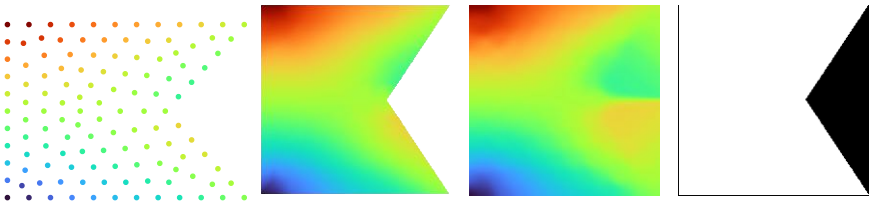


Fig. 3: **Constructing grid values from mesh values** (a) mesh nodes of a single output, the color of each node represents the stress value at the corresponding node, (b) reconstruction from Matplotlib `tricontourf` function, (c) our reconstruction on a 200×200 grid, (d) corresponding mask showing interpolated regions.

5 Experiments and Results

5.1 Data Generation

Gusset plates connect beams and columns to braces in steel structures. The behavior and analysis of these components are critical since various reports have observed failures of gusset plates subject to lateral loads [50–52]. The boundary conditions and time-history load cases are considered to simulate similar conditions in common gusset plate structures under external loading. Some of the most common gusset plate configurations in practice are shown in Fig 4.

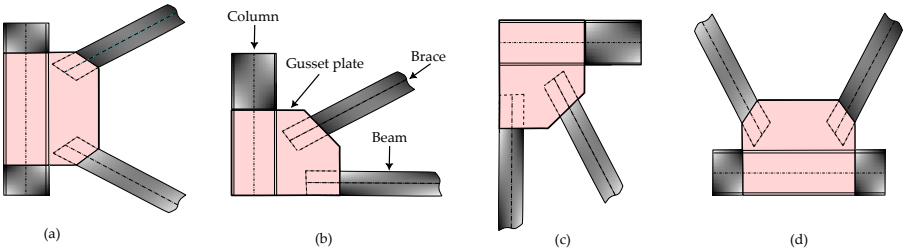


Fig. 4: Some of the most common gusset plates in practice.

We create a dataset with 71,680 unique samples by combining 14 random time-history load cases, 1024 different geometries, and 5 most commonly found boundary conditions in gusset plates. Boundary conditions are shown in Fig. 5, mimicking the real gusset plates' boundary conditions. All the translation and rotational displacements were fixed at the boundary conditions. The range for width and height of the plates is from 30 cm to 60 cm. Two-dimensional steel plate structures with five edges, E1 to E5 denoting edges 1 to 5, as shown in Fig. 6, are considered to be made of homogeneous and isotropic linear elastic

Table 2: Dataset splits

Split	Boundary condition	Load position	Load number	Geometry number
train	E2	E4E5	1-8	1-614
train	E2E3	E5	1-8	1-614
train	E1E2	E4	1-8	1-614
val	E3	E2E4	9-12	615-819
test	E1E5	E2	12-14	820-1024

materials. Various geometries are generated by changing the position of each node in horizontal and vertical directions, as shown in Fig. 6, which leads to 1024 unique pentagons. The material properties remain unchanged and isotropic for all samples.

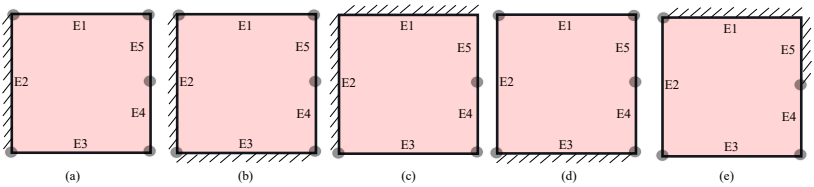


Fig. 5: Different types of boundary conditions for initializing population.

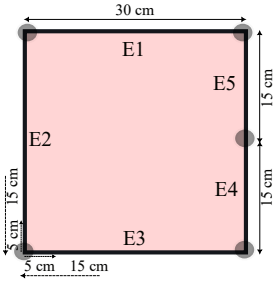
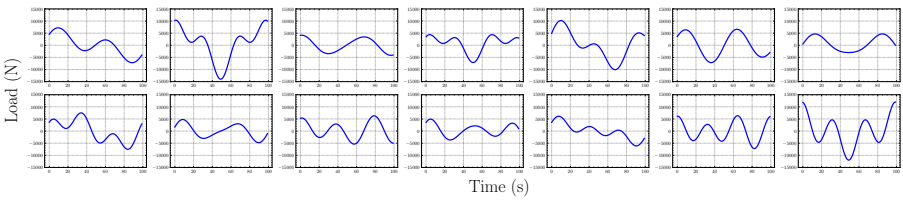
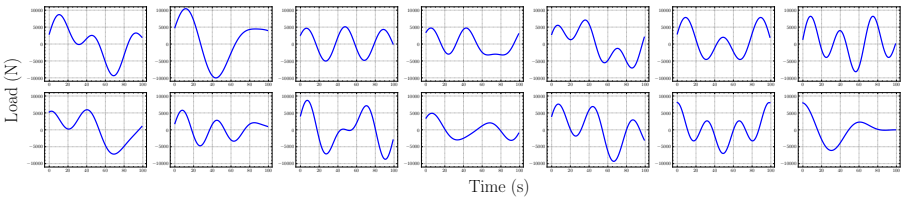


Fig. 6: Basic schematic topology for initializing the steel plate geometries.

Time histories consist of 100 time-steps generated with random sine and cosine frequencies. The frequencies range between 1 and 3 Hz, with amplitudes ranging from 2 to 10 kN at intervals of 2 kN. All time histories in horizontal and vertical directions are shown in Fig. 7. Each time series last for 1 second with each time-step lasting 0.01 seconds. All the details of the input variables used to initialize train-validation-test distribution of the population are shown in Table 2.



(a)



(b)

Fig. 7: Various load sequences in (a) horizontal and (b) vertical directions.

5.1.1 Input data

Input parameters include geometry, boundary condition, and body force in horizontal and vertical directions, each encoded as vectors in a 3-dimensional matrix. The size of the input matrix is $N \times M \times T$. where, N, M and T represent mesh nodes, input parameters and time, respectively. For example, if a sample contains 200 mesh nodes, the size of the input matrix is $200 \times 5 \times 100$. Fig. 8 shows how we construct the input matrix based on the geometry, boundary conditions and body forces. This figure presents a sample with five mesh nodes. However, all real samples in the trained model have more than 100 mesh nodes. The first and the second columns of the input matrix are x and y coordinates of the mesh nodes respectively. The third column represents the condition of boundary constraint at each node using a Boolean value. If there is a boundary constraint at the corresponding node, then the value is one, otherwise is zero. The fourth and the fifth columns represent body force sequences at each node along x and y directions. Details of boundary conditions and their load positions are described in Table 2.

5.1.2 Output Data

To obtain the stress distributions for each sample, we perform FEA using the Partial Differential Equation (PDE) solver in the MATLAB toolbox. Specifically, we use `transient-planestress` function of MATLAB PDE solver to generate dynamic stress contours which will act as the ground truth for our model. We define geometry, boundary condition, material properties, and time histories as input, and the PDE solver returns the sequence of stress distributions of σ_{xx}, σ_{yy} and σ_{xy} corresponding to the inputs. The size of each output is mesh nodes $\times$ load sequence. For example, if a sample contains 200 mesh

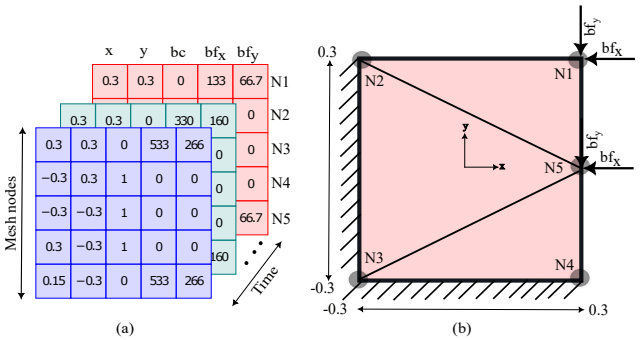


Fig. 8: Construction of input matrix (Unit: m, N).

nodes, the size of the output matrix is 200 × 100. Each of the three outputs are normalized separately between -1 and 1 to ensure faster convergence. The input and the output representations of the model is shown in Fig. 9.

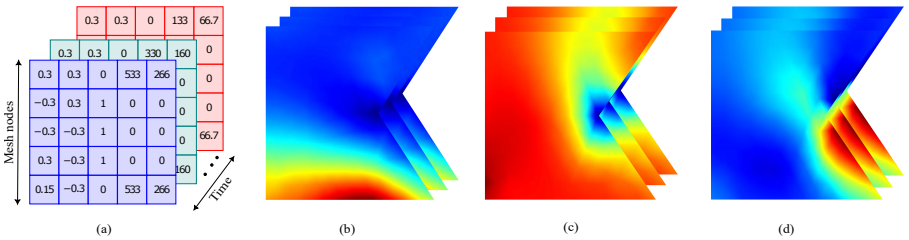


Fig. 9: Input and output representation for normal and shear stress distribution prediction: (a) Input matrix, (b) Output (σ_{xx}), (c) Output (σ_{yy}), (d) Output (σ_{xy}).

5.2 Metrics

We use Mean Absolute Error (MAE), defined in Eq. 10 as the primary training loss and metric. To ensure that we do not overfit to a single metric, we also use Mean Relative Percentage Error (MRPE) to evaluate the overall quality of predicted stress distribution.

$$\text{MAE} = \frac{1}{NT} \sum_{N,T}^{n,t} |S(n,t) - \hat{S}(n,t)| \tag{10}$$

$$\text{MRPE} = \frac{\text{MAE}}{\max |S(n,t), \hat{S}(n,t)|} \times 100 \tag{11}$$

where $S(n,t)$ is the true stress value at a node n at time step t, as computed by FEA, and $\hat{S}(n,t)$ is the corresponding stress value predicted by our model, N is the total number of mesh nodes in each frame of a sample, and T is a total

number of time steps in each sample. As mentioned earlier, we set $T = 100$ in our experiments.

5.3 Implementation

We implemented our model using PyTorch [53] and PyTorch Lightning. AdamW optimizer [48] was used with an initial learning rate of 10^{-3}. All the details of the network hyperparameters can be found in table 3. The computational performance of the model was evaluated on an AMD EPYC 7313 16-core processor and one NVIDIA A6000 48GB GPU per experiment. The time required during the training phase for a single batch with 100 frames and a batch size of 10 for NeuroStress and PINN-Stress were 7 and 20 milliseconds respectively. The inference time of NeuroStress and PINN-Stress for one sample was 1 millisecond. The most powerful FE solvers take between 2 minutes to an hour to solve the same. We use MATLAB PDE solver as a FE solver to compare the efficiency of our model. We assumed that one sample takes about two minutes to solve in the FE solver, regardless of how much modeling there is in the FE solver. MATLAB PDE solver does not use GPU acceleration. Therefore, NeuroStress and PINN-Stress in solution time are about 12×10^4 times faster than MATLAB PDE solver.

Table 3: Network hyperparameters

Batch size	Learning rate	Weight decay	Number of STM modules	Loss functions
10	1e−3	1e−4	3	MAE-Phy

5.4 Results

We implement two main models, NeuroStress and PINN-Stress. Both models are trained on the same train dataset for 300 epochs, evaluated on the validation dataset for fine-tuning, and we report all metrics on the test dataset. The entire dataset contains 71,680 samples, while the train dataset contains 43,008 samples, validation and test datasets each contain 14336, forming the 60%-20%-20% split of the whole dataset. Error metrics are calculated using the checkpoint with the least validation error. Fig. 10 shows stress distribution prediction for σ_{xx}, σ_{yy}, σ_{xy} and σ_{vm} of a randomly selected frame in a sample. PINN-Stress predictions are almost identical to their corresponding references, and the errors in a PINN-Stress prediction are substantially lower than those in a NeuroStress prediction. Particularly, PINN-Stress can capture peak stress better than NeuroStress, which is of primary importance in structural design. The importance of maximum stress matters in the design phase since maximum stress should be less than yield strength to avoid permanent deformation.

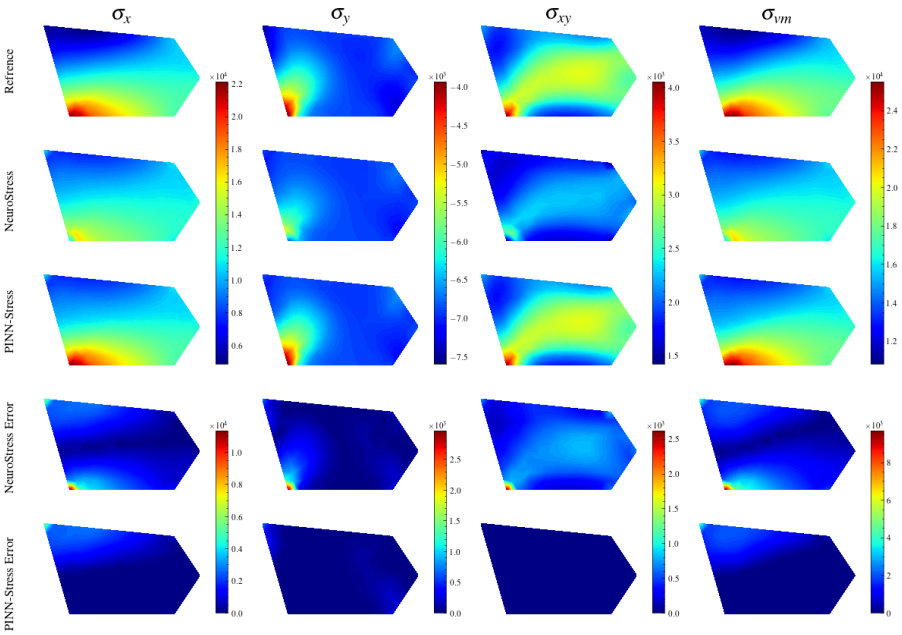


Fig. 10: Comparison of NeuroStress and PINN-Stress predictions for σ_{xx}, σ_{yy}, σ_{xy} and σ_{vm}. (Unit: MPa)

6 Ablation Studies

6.1 Generalization

We investigate and compare the generalization capabilities of NeuroStress and PINN-Stress models for varying distributions of boundary conditions, load sequences and geometries. To that end, we collect the entire dataset and split them into train, validation and test sets such that validation and test sets contain unseen instances of the entity to check generalization on. For example, for checking generalization on geometry, train set will consist of 614 geometries out of 1024, and validation and test sets will contain the remaining (205 each). Table 4 represents the data split for the generalization experiment.

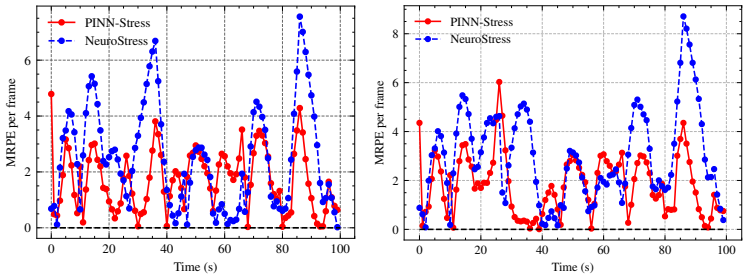
We compare the mean relative percent error (MRPE) of each method on von Mises stress prediction. As von Mises stress identifies if a given material is likely to yield or fracture, we use its prediction error as the sole criterion. Fig 11 shows the relative error of each frame for a random sample across all time frames for unseen load sequences and structural geometries. As can be seen, in both figures, the relative errors in PINN-Stress are less than NeuroStress, especially in extreme peaks. For instance, in the last two peaks of the Figs 11a and 11b the relative error of the PINN-Stress is around 10% less than NeuroStress. This demonstrates the ability of PINN-Stress to predict the maximum stress values. Additionally, the relative error of PINN-Stress in both

Table 4: Data split for generalization experiments

Quantity	Data split*			MRPE (%)	
	Train	Val	Test	NeuroStress	PINN-Stress
Geometry	1-614	615-819	820-1024	1.7	**1.5**
Load	1-8	9-11	12-14	4.8	**4.2**
BC	E2, E2E3, E1E2	E3	E1E5	18.3	**16**

* The values in the data split column refer to indices of the corresponding generalization quantity.

unseen load sequences and structural geometries for almost all 100 frames is less than 4%, and at least half of the frames have a relative error of less than 2% which we deem acceptable in the engineering domain.



(a) (b)

Fig. 11: Comparison of NeuroStress and PINN-Stress errors for σ_{vm} across 100 frames for a random spatial node in a sample. (a) unseen load sequences and (b) unseen geometries.

We pick a random frame from a randomly selected sample to visualize the stress distribution for unseen load sequences as shown in Fig 12. This figure demonstrates the generalization capability of PINN-Stress and NeuroStress to unseen load sequences. As it can be seen, σ_{xx}, σ_{yy}, σ_{xy} and σ_{vm} predictions by PINN-Stress are significantly better than those by NeuroStress. Particularly, the PINN-Stress is more effective at capturing extreme peak values than the NeuroStress.

We have also compared the generalization capability of PINN-Stress and NeuroStress over unseen load sequences in a single spatial node across all time frames in Fig 13. Fig 13 demonstrate the ability of our models to capture the temporal dependencies over time frames. It can be seen that both models' predictions are almost identical to references in all the time frames. However, in extreme peaks PINN-Stress outperforms NeuroStress.

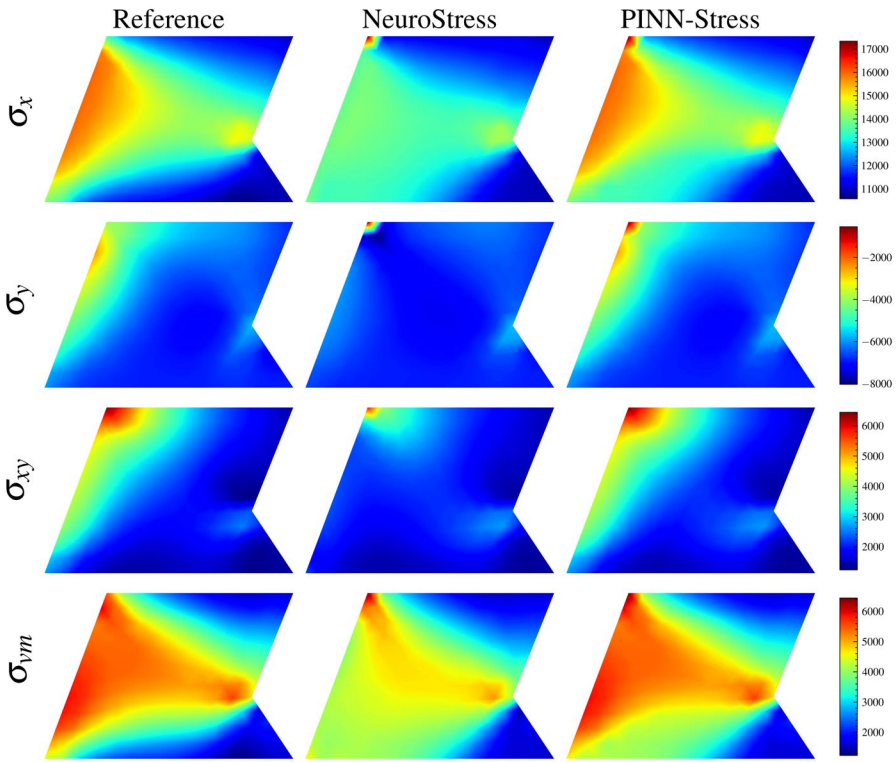


Fig. 12: Predicting dynamic stress distribution for diverse load sequences (Unit: MPa).

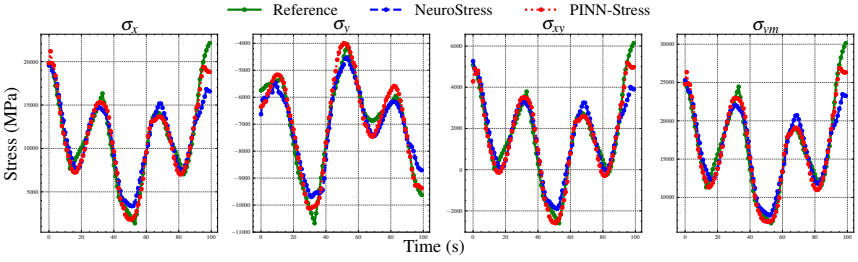


Fig. 13: Comparison of NeuroStress and PINN-Stress predictions for σ_{xx}, σ_{yy}, σ_{xy} and σ_{vm} across 100 frames for a sample with unseen load sequences.

Fig 14 demonstrates the generalization capability of PINN-Stress and NeuroStress to unseen geometries for a random frame from a sample. As it can be seen, σ_{xx}, σ_{yy}, σ_{xy} and σ_{vm} predictions by PINN-Stress are significantly better than those by NeuroStress. The NeuroStress can just predict the overall pattern of stress distribution; however, the PINN-Stress can predict every details of stress distribution. Fig 15 shows stress values in a single spatial node across

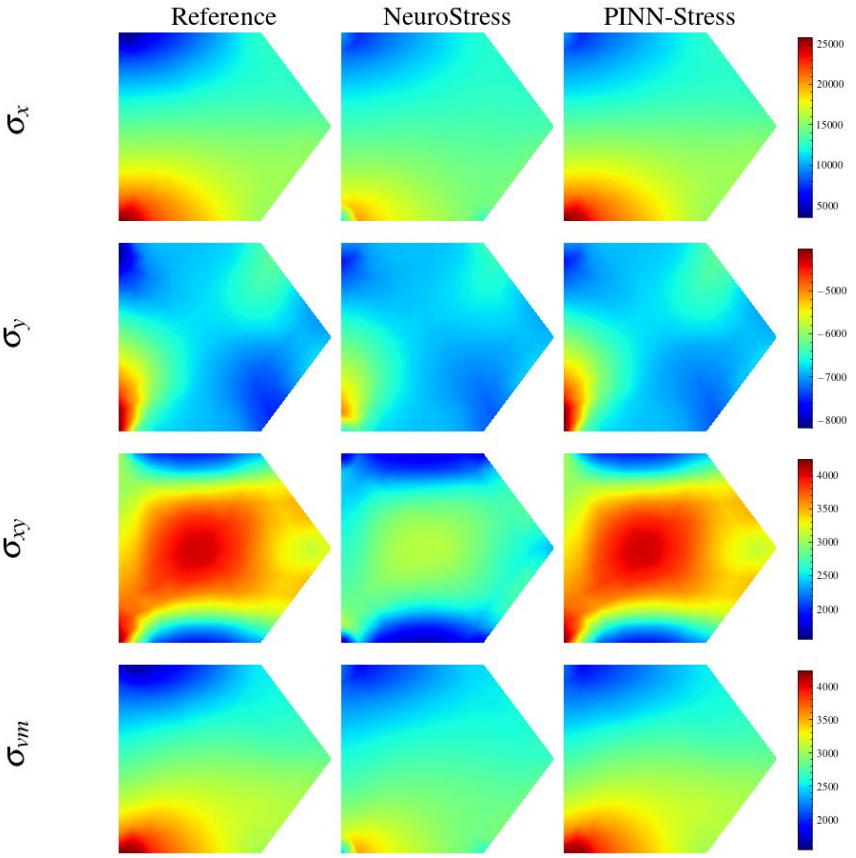


Fig. 14: Predicting dynamic stress distribution for diverse geometries (Unit: MPa).

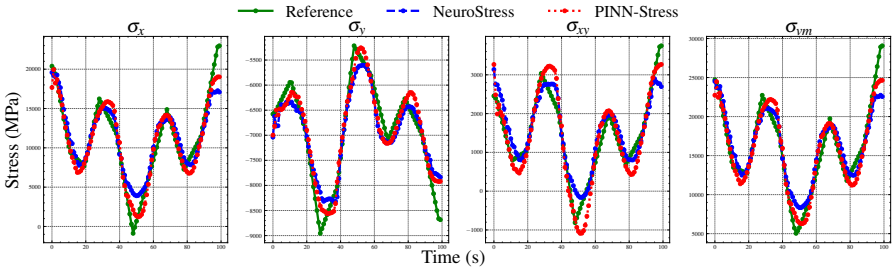


Fig. 15: Comparison of NeuroStress and PINN-Stress predictions for σ_{xx}, σ_{yy}, σ_{xy} and σ_{vm} across 100 frames for a sample with unseen geometries.

all time frames for the randomly selected sample with unseen geometries. predictions of PINN-Stress in unseen geometries like unseen load sequences are almost identical to the references and, in peak locations, are more fitted to the

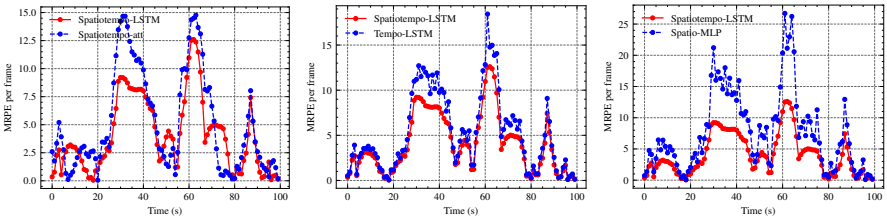
references compared to the NeuroStress. The results of geometries generalization can be found in Table 4. The lowest error in each experiment is highlighted in bold. In every experiment, we can observe that PINN-Stress generalizes better than NeuroStress. However, neither method generalizes satisfyingly for various boundary conditions. Since we only considered five different boundary conditions in total, we ran the same experiment for different combinations of boundary conditions and the results were similar.

6.2 Choice of architecture

The efficiency of architecture can be attributed to several design choices we have made. Our architecture models the temporal dependency between time frames and the relationship between different nodes in an input via our spatiotemporal multiplexing mechanism. As mentioned earlier, we are the first to introduce such a design into PINNs to the best of our knowledge. Even though self-attention has shown state-of-the-art performance in sequence modeling, they are not suitable for tasks without large amounts of data. Hence, we use LSTMs for sequence modeling. To demonstrate our claim, we compare our architecture against other baseline architectures. We compare against three architectures: **Spatiotempo-Att**, **Tempo-LSTM**, **Spatio-MLP**. Spatiotempo-Att is very similar to our architecture, except the LSTM modules in our model are replaced with self-attention modules. Tempo-LSTM is also similar to our architecture except the LSTMs act only along the temporal dimension. Spatio-MLP is a normal feedforward network with six layers with LeakyReLU activation in between. It treats each time frame separately but considers all the nodes simultaneously. We will refer to our architecture as **Spatiotempo-LSTM**. To save time and resources, we train all the architectures on 10% of training data with MAE loss. Similar to our experiments on generalization, we report the error on von Mises stress prediction. The results are shown in Table 5, and the best results are highlighted in bold Fig 16 shows the relative error of each frame for a random sample across all time frames for baseline architectures compared to our architecture. As it can be seen Spatiotempo-LSTM has less relative error in most of the frames compared to the baseline architectures.

Table 5: Architecture comparison

	Architecture			
	Spatiotempo-Att	Tempo-LSTM	Spatio-MLP	Spatiotempo-LSTM
#Params (K)	309	208	828	208
MRPE(%)	19.5	17.5	25.4	**16.6**



(a) (b) (c)

Fig. 16: Comparison of baselines architectures and our architecture for σ_{vm} across 100 frames for a random spatial node in a sample. (a) Spatiotempo-LSTM vs Spatiotempo-att, (b) Spatiotempo-LSTM vs Tempo-LSTM and (c) Spatiotempo-LSTM vs Spatio-MLP

.

6.3 Baselines

We selected two state-of-the-art methods as the baselines for comparison. In the first method proposed by Nie et al. [16] U-net architecture was employed to predict the static stress distribution of cantilever beams. Specifically, they used conv and deconv layers as encoder-decoder and res-net and squeeze and excitation blocks in the latent space. We changed the input matrix which was discussed in section 5.1.2 into images same as our paper [1] in order to be compatible with CNN. Since their architecture was designed for static cases and our dataset was sequential we passed each 100 frames separately into their model. For the second baseline, we used Neuro-DynaStress proposed by [1] which was employed to predict dynamic stress distribution utilizing CNN and LSTM. The architecture contains conv and deconv layers as encoder-decoder, Feature alignment modules that directly pass information from encoder to decoder, and LSTM blocks in the latent space. The results are shown in Table 6, and the best results are highlighted in bold. PINN-Stress outperforms all other methods when it comes to accuracy, while NeuroStress proves to be quicker than other methods.

Table 6: Baselines comparison

	Architecture (MRPE(%))			
Quantity	PINN-Stress	NeuroStress	StressNet	Neuro-DynaStress
Geo	**1.5**	1.7	5.2	2.5
Load	**4.2**	4.8	7.8	5.6
BC	**16**	18.3	25.4	20.1
Training time per sample (ms)	20	**7**	1500	800

7 Conclusion

We propose NeuroStress and PINN-Stress, two models for dynamic stress prediction based on a novel architecture, with the latter augmented with physics-informed loss function. Our models explicitly learn both spatial and temporal information through our spatiotemporal multiplexing (STM) module. Experiments on simulated gusset plates show that not only are our models accurate, but adding physics-informed loss function facilitates generalization with respect to varying load sequences and structural geometries. PINN-Stress is also better at estimating high stress values which is of more importance to the structural engineering community. However, collecting sufficient data points from real gusset plates using sensors can be expensive and noisy. Therefore, our future efforts will be directed toward achieving lower sample complexity under noisy conditions.

Declarations

- This research was funded in part by the National Science Foundation grant CNS 1645783.
- There is no conflict of interest among the authors of this paper
- The dataset and codes generated and/or analyzed during the current study are available at GitHub

References

[1] Bolandi, H., Li, X., Salem, T., Boddeti, V., Lajnef, N.: Bridging finite element and deep learning: High-resolution stress distribution prediction in structural components. Frontiers of Structural and Civil Engineering (2022)

[2] Bolandi, H., Li, X., Salem, T., Boddeti, V., Lajnef, N.: Deep learning paradigm for prediction of stress distribution in damaged structural components with stress concentrations. Advances in Engineering Software **173**, 103240 (2022)

[3] Raissi, M., Perdikaris, P., Karniadakis, G.E.: Machine learning of linear differential equations using gaussian processes. Journal of Computational Physics **348**, 683–693 (2017)

[4] Raissi, M., Perdikaris, P., Karniadakis, G.E.: Numerical gaussian processes for time-dependent and nonlinear partial differential equations. SIAM Journal on Scientific Computing **40**(1), 172–198 (2018)

[5] Raissi, M., Wang, Z., Triantafyllou, M.S., Karniadakis, G.E.: Deep learning of vortex-induced vibrations. Journal of Fluid Mechanics **861**, 119–137 (2019)

[6] Astaneh-Asl, A.: Gusset plates in steel bridges–design and evaluation. Steel TIPS report, structural steel educational council technical information & product services. Moraga, CA (2010)

[7] Farimani, A.B., Gomes, J., Pande, V.S.: Deep learning the physics of transport phenomena. arXiv preprint arXiv:1709.02432 (2017)

[8] Kim, B., Azevedo, V.C., Thuerey, N., Kim, T., Gross, M., Solenthaler, B.: Deep fluids: A generative network for parameterized fluid simulations. In: Computer Graphics Forum, vol. 38, pp. 59–70 (2019). Wiley Online Library

[9] Goh, G.B., Hodas, N.O., Vishnu, A.: Deep learning for computational chemistry. Journal of computational chemistry **38**(16), 1291–1307 (2017)

[10] Mardt, A., Pasquali, L., Wu, H., Noé, F.: Vampnets for deep learning of molecular kinetics. Nature communications **9**(1), 1–11 (2018)

[11] Mohammadi Bayazidi, A., Wang, G.-G., Bolandi, H., Alavi, A.H., Gandomi, A.H.: Multigene genetic programming for estimation of elastic modulus of concrete. Mathematical Problems in Engineering **2014** (2014)

[12] Sarveghadi, M., Gandomi, A.H., Bolandi, H., Alavi, A.H.: Development of prediction models for shear strength of sfrcb using a machine learning approach. Neural Computing and Applications **31**(7), 2085–2094 (2019)

[13] Mousavi, S.M., Aminian, P., Gandomi, A.H., Alavi, A.H., Bolandi, H.: A new predictive model for compressive strength of hpc using gene expression programming. Advances in Engineering Software **45**(1), 105–114 (2012)

[14] Bolandi, H., Banzhaf, W., Lajnef, N., Barri, K., Alavi, A.H.: An intelligent model for the prediction of bond strength of frp bars in concrete: A soft computing approach. Technologies **7**(2), 42 (2019)

[15] Modarres, C., Astorga, N., Droguett, E.L., Meruane, V.: Convolutional neural networks for automated damage recognition and damage type identification. Structural Control and Health Monitoring **25**(10), 2230 (2018)

[16] Nie, Z., Jiang, H., Kara, L.B.: Stress field prediction in cantilevered structures using convolutional neural networks. Journal of Computing and Information Science in Engineering **20**(1), 011002 (2020)

[17] Jiang, H., Nie, Z., Yeo, R., Farimani, A.B., Kara, L.B.: Stressgan: A generative deep learning model for two-dimensional stress distribution prediction. Journal of Applied Mechanics **88**(5) (2021)

[18] Zhang, R., Chen, Z., Chen, S., Zheng, J., Büyüköztürk, O., Sun, H.: Deep long short-term memory networks for nonlinear structural seismic response prediction. Computers & Structures **220**, 55–68 (2019)

[19] Do, D.T., Lee, J., Nguyen-Xuan, H.: Fast evaluation of crack growth path using time series forecasting. Engineering Fracture Mechanics **218**, 106567 (2019)

[20] Presas, A., Valentin, D., Zhao, W., Egusquiza, M., Valero, C., Egusquiza, E.: On the use of neural networks for dynamic stress prediction in francis turbines by means of stationary sensors. Renewable Energy **170**, 652–660 (2021)

[21] Raissi, M., Perdikaris, P., Karniadakis, G.E.: Inferring solutions of differential equations using noisy multi-fidelity data. Journal of Computational Physics **335**, 736–746 (2017)

[22] Raissi, M., Yazdani, A., Karniadakis, G.E.: Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science **367**(6481), 1026–1030 (2020)

[23] Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561 (2017)

[24] 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. Journal of Computational physics **378**, 686–707 (2019)

[25] Vahab, M., Haghighat, E., Khaleghi, M., Khalili, N.: A physics-informed neural network approach to solution and identification of biharmonic equations of elasticity. Journal of Engineering Mechanics **148**(2), 04021154 (2022)

[26] Yan, C., Vescovini, R., Dozio, L.: A framework based on physics-informed neural networks and extreme learning for the analysis of composite structures. Computers & Structures **265**, 106761 (2022)

[27] Chen, D., Li, Y., Liu, K., Li, Y.: A physics-informed neural network approach to fatigue life prediction using small quantity of samples. International Journal of Fatigue **166**, 107270 (2023)

[28] Bai, J., Rabczuk, T., Gupta, A., Alzubaidi, L., Gu, Y.: A physics-informed neural network technique based on a modified loss function for computational 2d and 3d solid mechanics. Computational Mechanics **71**(3), 543–562 (2023)

[29] Jeong, H., Bai, J., Batuwatta-Gamage, C., Rathnayaka, C., Zhou, Y., Gu, Y.: A physics-informed neural network-based topology optimization (pinnto) framework for structural optimization. Engineering Structures **278**, 115484 (2023)

[30] Zhang, E., Dao, M., Karniadakis, G.E., Suresh, S.: Analyses of internal structures and defects in materials using physics-informed neural networks. Science advances **8**(7), 0644 (2022)

[31] Fallah, A., Aghdam, M.M.: Physics-informed neural network for bending and free vibration analysis of three-dimensional functionally graded porous beam resting on elastic foundation. Engineering with Computers, 1–18 (2023)

[32] Bazmara, M., Silani, M., Mianroodi, M., *et al.*: Physics-informed neural networks for nonlinear bending of 3d functionally graded beam. In: Structures, vol. 49, pp. 152–162 (2023). Elsevier

[33] Zhao, X., Gong, Z., Zhang, Y., Yao, W., Chen, X.: Physics-informed convolutional neural networks for temperature field prediction of heat source layout without labeled data. Engineering Applications of Artificial Intelligence **117**, 105516 (2023)

[34] Xu, C., Cao, B.T., Yuan, Y., Meschke, G.: Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios. Computer Methods in Applied Mechanics and Engineering **405**, 115852 (2023)

[35] Zheng, B., Li, T., Qi, H., Gao, L., Liu, X., Yuan, L.: Physics-informed machine learning model for computational fracture of quasi-brittle materials without labelled data. International Journal of Mechanical Sciences **223**, 107282 (2022)

[36] Yao, H., Gao, Y., Liu, Y.: Fea-net: A physics-guided data-driven model for efficient mechanical response prediction. Computer Methods in Applied Mechanics and Engineering **363**, 112892 (2020)

[37] Das, S., Dutta, S., Putcha, C., Majumdar, S., Adak, D.: A data-driven physics-informed method for prognosis of infrastructure systems: Theory and application to crack prediction. ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering **6**(2), 04020013 (2020)

[38] Wang, R., Kashinath, K., Mustafa, M., Albert, A., Yu, R.: Towards physics-informed deep learning for turbulent flow prediction. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1457–1466 (2020)

[39] Goswami, S., Yin, M., Yu, Y., Karniadakis, G.E.: A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. Computer Methods in Applied Mechanics and Engineering **391**, 114587 (2022)

[40] Haghighat, E., Raissi, M., Moure, A., Gomez, H., Juanes, R.: A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Computer Methods in Applied Mechanics and Engineering **379**, 113741 (2021)

[41] Jin, X., Cai, S., Li, H., Karniadakis, G.E.: Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. Journal of Computational Physics **426**, 109951 (2021)

[42] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895 (2020)

[43] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[44] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[45] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. A field guide to dynamical recurrent neural networks. IEEE Press In (2001)

[46] Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE **78**(10), 1550–1560 (1990)

[47] Zeyer, A., Bahar, P., Irie, K., Schlüter, R., Ney, H.: A comparison of transformer and lstm encoder decoder models for asr. In: 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 8–15 (2019). IEEE

[48] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)

[49] Ke, N.R., Chiappa, S., Wang, J., Bornschein, J., Weber, T., Goyal, A., Botvinic, M., Mozer, M., Rezende, D.J.: Learning to induce causal structure. arXiv preprint arXiv:2204.04875 (2022)

[50] ZAHRAEI, S.M., Heidarzadeh, M.: Destructive effects of the 2003 bam earthquake on structures (2007)

[51] Zahrai, S.M., Bolandi, H.: Towards lateral performance of cbf with unwanted eccentric connection: A finite element modeling approach. KSCE Journal of Civil Engineering **18**(5), 1421–1428 (2014)

[52] Zahrai, S., Bolandi, H.: Numerical study on the impact of out-of-plane eccentricity on lateral behavior of concentrically braced frames. International Journal of Steel Structures **19**(2), 341–350 (2019)

[53] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)