BEYOND FINITE ELEMENT: PHYSICS INFORMED NEURAL NETWORK FOR STRESS PREDICTION

By

Hamed Bolandi

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Civil Engineering—Doctor of Philosophy
Computer Science—Dual Major

2023

ABSTRACT

This Multidisciplinary research proposes deep neural networks to bypass the Finite Element Analysis (FEA) and predict high-resolution stress distributions on loaded steel plates with variable loading, geometries, and boundary conditions. FEA for structures has been broadly used to conduct stress analysis of various civil and mechanical engineering structures. Conventional methods, such as FEA, provide high-fidelity solutions but require solving large linear systems that can be computationally intensive. The existing workflow for FEM applications includes: (i) modeling the geometry and its components, (ii) specifying material properties, boundary conditions, and loading, (iii) Applying mesh strategy, and (iv) stress analysis which may be time-consuming based on the complexity of the model. Instead, Deep learning (DL) techniques can generate solutions significantly faster than conventional run-time analysis. This can prove extremely valuable in real-time structural assessment applications. In this work, The Convolutional Neural network (CNN) was designed and trained to use the geometry, boundary conditions, and static load as input to predict the stress contours in intact steel plates. Furthermore, we predict high-resolution stress distributions on damaged steel plates using CNNs augmented with custom loss functions that use physics rules to bypass the need for Finite Element Analysis. We embedded physics constraints into the loss function to enforce the model training, precisely capturing stress concentrations around the tips of various structural damage configurations. The proposed technique's performance was compared to Finite-Element simulations using partial differential equation (PDE) solvers. Neuro-DynaStress is also proposed to predict the entire sequence of stress distribution based on Finite Element simulations using a partial differential equation (PDE) solver. More specifically, CNN, along with the multi-head attention transformer and feature alignment, is used to extract features and capture the data's temporal dependence. The model was designed and trained to use the geometry, boundary conditions, and sequence of loads as input and predict the sequences of high-resolution von Mises stress contours. Moreover, to increase the accuracy of dynamic stress prediction, we propose a Physics Informed Neural Network (PINN). The PINN-Stress model can predict the entire sequence of stress distribution based on finite element simulations using a PDE solver. In order

to force our model to learn the physical constraints, we minimize the violation of the equation of motion and also minimize the boundary condition violation to fully enforce the underlying PDE. The PINN-Stress model can predict the sequence of normal and shear stress distribution in almost real-time and can generalize better than the model without PINN. Our model is also able to predict von Mises stress using the von Mises equation.

ACKNOWLEDGMENTS

Words cannot express my gratitude to my advisors, Dr. Nizar Lajnef and Dr. Vishnu Boddeti, for their invaluable patience and feedback. I could not have undertaken this journey without them, who generously provided knowledge and expertise. Additionally, this endeavor would not have been possible without the support of my committee, Dr. Wolfgang Banzhaf and Dr. Weiyi Lu.

I am also grateful to my friends and lab mates, Xuyang Li, and Bashir Sadeghi, for their valuable comments and especially to Gautam Sreekumar, for his valuable guidance and help. I would like to thank Laura Post for making the paperwork process so easy and for her help with the administrative process.

Last but not least, I would be remiss in not mentioning my family; thanks to my wife Samane for her understanding and love during the past few years. Her support and encouragement were in the end what made this dissertation possible. My parents, Alireza and Zohre, receive my deepest gratitude and love for their dedication and the many years of support during my undergraduate studies that provided the foundation for this work.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background and State of Knowledge

Stress analysis is an essential part of engineering and design. The development of various design systems continuously imposes higher demands on computational costs while preserving accuracy. Numerical analysis methods, such as structural Finite Element Analysis (FEA), are typically used to conduct stress analysis of various structures. Researchers commonly use FEA methods to evaluate the design, safety, and maintenance of different structures in various fields, including aerospace, automotive, architecture, civil, and structural systems. The current workflow for FEA applications includes: a) modeling geometry and its components, which can be time-consuming based on the system complexity; b) specifying material properties, boundary conditions, and loading; c) applying a meshing strategy for geometry; d) analyzing, which complexity of all previous steps determines how long does it take. The time consumption and complexity of current FEA workflows make it impractical in real-time or near real-time applications, such as in the aftermath of a disaster or during extreme disruptive events that require immediate corrections to avoid catastrophic failures. Based on the steps of FEA described above, performing a complete stress analysis with conventional FEM has high computational costs. To resolve this issue, we propose Deep Learning (DL) methods to construct deep neural networks (DNN), which, once trained, allow to bypass FEA. These methods may enable real-time stress analysis by leveraging machine learning (ML) algorithms. DNNs can model complicated, nonlinear relationships between input and output data. Thus, these models help us acquire adequate knowledge for predictions of unseen problems.

Data-driven approaches that model physical phenomena have been lauded for their significant and growing successes. Most recent works have included design, and topology optimization [1, 2], data-driven approaches in fluid dynamics [3, 4], molecular dynamics simulation [5, 6], and material properties prediction [7, 8, 9, 10]. Atalla et al. [11] and Levin et al. [12] have used neural

regression for FEA model updating. Recently, DL has shown promise in solving conventional mechanics problems. Some researchers used DL for structural damage detection, a promising alternative to conventional structural health monitoring methods [13, 14]. Javadi et al. [15] used a typical neural network in FEA as a surrogate for the traditional constitutive material model. They simplified the geometry into a feature vector which approaches hard to generalize complicated cases. The numerical quadrature of the element stiffness matrix in the FEA on a per-element basis was optimized by Oishi et al. [16] using deep learning. Their approach helps to accelerate the calculation of the element stiffness matrix. Convolutional Neural Networks (CNN) are commonly used in tasks involving 2D information due to the design of their architecture. Recently, Madani et al. [17] developed a CNN architecture for stress prediction of arterial walls in arteriosclerosis. Also, Liang et al. [18] proposed a CNN model for aortic wall stress prediction. Their method is expected to allow real-time stress analysis of human organs for a wide range of clinical applications.

Gulgec et al. [19] proposed a CNN architecture to classify simulated damaged and intact samples and localize the damage in steel gusset plates. Modares et al. [20] conducted a study on composite materials to identify the presence and type of structural damage using convolutional neural networks. Also, for detecting concrete cracks without calculating the defect features, Cha et al. [21] proposed a vision-based method based on convolutional neural networks (CNNs). Do et al. [22] proposed a method for forecasting the crack propagation in risk assessment of engineering structures based on “long short-term memory" and “multi-layer neural network". Truong et al. [23] proposed a deep forward neural network method to detect the location and severity of damaged elements in the bar planar truss and bar dome-like space truss using the noisy incomplete modal dat. Lieu et al. [24] presented a deep neural network-based adaptive surrogate model for structural reliability analysis. Zhuang et al. [25] developed a technique for bending, vibration, and buckling analysis of Kirchhoff plates based on deep autoencoders. Samaniego et al. [26] proposed a deep neural network to solve boundary value problems. They used relevant examples, from computational mechanics, using DNNs to build the approximation space. A deep feed-forward artificial neural network has been developed by Berg et al. [27] to approximate partial differential equations with complex geometry.

Truong et al. [28] proposed a method for the safety evaluation of steel trusses using the gradient tree boosting algorithm.

An approach for predicting stress distribution on all layers of non-uniform 3D parts was presented by Khadilkar et al. [29] More recently, Nie et al. [30] developed a CNN-based method to predict the low-resolution stress field in a 2D linear cantilever beam. Jiang et al. [31] developed a conditional generative adversarial network for low-resolution von Mises stress distribution prediction in solid structures. Some studies have been conducted to develop methods of predicting structural response using ML models. Dong et al. [32] proposed a support vector machine approach to predict nonlinear structural responses. Wu et al. [33] Utilized deep convolutional neural networks to estimate the structural dynamic responses. Long short-term memory (LSTM) [34] was used by Zhang et al. [35] to predict nonlinear structural response under earthquake loading. Fang et al. [36] proposed a deep-learning-based structural health monitoring (SHM) framework capable of predicting a dam's structural dynamic responses once explosions are experienced using LSTM. Kohar et al. [37] used 3D-CNN-autoencoder and LSTM to predict the force-displacement response and deformation of the mesh in vehicle crash-worthiness. Schwarzer et al. [38] construct a neural network architecture that combines a graph convolutional neural network (GCN) with a recurrent neural network (RNN) to predict fracture propagation in brittle materials. Lazzara et al. [39] proposed a dual-phase LSTM Auto-encoder-based surrogate model to predict aircraft dynamic landing response over time. Jahanbakht et al. [40] presented an FEA-inspired DNN using an attention transformer to predict the sediment distribution in the wide coral reef.

The few models that studied stress predictions suffer from the problem of low-resolution predictions and ungeneralizability, making them unsuitable for decision-making after a catastrophic failure. To the best of our knowledge, this is the first work to predict stress distribution in the specific domain of steel plates with high accuracy and low latency.

1.2 Research Objectives

Based on the steps of FEA described in the last section, performing a complete stress analysis with conventional FEA has a high computational cost. The main objective of this work is to bypass

FEA to provide faster FEA for engineering communities. Our approach in a sense, could be viewed as a surrogate for FEA software, and it avoids the computations bottlenecks in FEA. In particular, our model predicts stress distributions and stress concentrations in both static and dynamic models of the most common gusset plates used in infrastructures such as bridges and buildings. The proposed deep learning models are established through the integration of AI (Artificial intelligence) and FE methods. Advanced AI algorithms are developed and evaluated during this research. The main idea here is to train a generalized model that can later be used in situations where real-time estimations are needed, such as in the aftermath of extreme disruptive events. For example, focusing on critical structural components, there is a need for immediate assessment following a disaster or during extremely disruptive events to guide corrective actions. Engineers could rely on the proposed computationally efficient algorithms to determine stress distributions over gusset plates and apply the proper rehabilitation actions. It is important for them to be able to analyze gusset plates quickly and accurately, which is exactly what our models can provide.

1.3 Research Significance

We propose deep learning models that can act as a surrogate for FEA solvers for dynamic FEA while avoiding the computational bottlenecks involved. To demonstrate its utility, we model the stress distribution in gusset plates under static and dynamic loading. Bridges and buildings rely heavily on gusset plates as one of their most critical components. Gusset plates are designed to withstand lateral loads such as earthquakes and winds, which makes fast dynamic models valuable in avoiding catastrophic failures. The main idea here is to train a model that can later be used when real-time estimations are needed, such as in the aftermath of extreme disruptive events. For example, focusing on critical structural components, there is a need for immediate assessment following a disaster or during extremely disruptive events to guide corrective actions. Engineers could rely on the proposed computationally efficient algorithms to determine stress distributions over damaged gusset plates and apply the proper rehabilitation actions. They need to be able to analyze gusset plates quickly and accurately, which is what our model can provide.

1.4 Contributions

1.4.1 Structural Engineering

We have made a major advancement in the field of Structural Engineering. For the first time, we have developed a deep learning model for the prediction of high-resolution stress distribution in intact structural components. Our model has achieved a Percentage Mean Absolute Error (PMAE) of 0.9% and a Percentage Peak Absolute Error (PPAE) of 0.46%. We also proposed a convolutional neural network (CNN) augmented with the custom loss function which is inspired by the stress concentration physics equation to predict high-resolution von Mises stress distribution in the specific domain of damaged steel plates. The custom loss helped to localize damages accurately and capture stress concentration around crack tips, which is not possible with other ML methods. Moreover, we built a novel deep neural network equipped with a Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) to predict the entire sequence of dynamic stress distribution. The model can predict dynamic stress distribution with a mean relative percentage error of 2.3%, which is considered an acceptable error rate in engineering communities. Furthermore, We presented a new model, called PINN-Stress, that can accurately predict dynamic stress distribution. Our model utilizes a unique architecture and incorporates a physics-informed loss function, resulting in a computational speed that is 600,000 times faster than traditional finite element solvers.

1.4.2 Computer Science

Presenting novel custom loss function using dynamic binary masks. The custom loss function is formulated by taking into account the stress concentration at the crack tips, which is the highest compared to other locations in the section. The proposed approach is able to localize the cracks and capture the high-resolution stress concentration at the crack tips in structural components, even when the crack length and width are too small. Propose a state-of-the-art physics-informed loss function for the first time that uses a governing equation behind the equation of motion as a soft constraint to enforce the loss to minimize. This loss function is computationally efficient and facilitates real-time analysis for dynamic stress distribution prediction. We also propose a novel

method to calculate gradients of stresses on a surrogate grid created using kernel density estimation (KDE). This method helped us to estimate the gradients of stress output along the x and y directions.

1.5 Chapter Overview

The study will be guided by four major tasks for bypassing FEA, by introducing novel deep-learning models. An overview of each chapter is shown below, delineating the specific objective, body, and conclusion.

Chapter II deals with developing deep neural networks in the form of convolutional neural networks (CNN) to bypass the FEA and predict high-resolution static stress distributions on loaded steel plates with variable geometry, loading, and boundary conditions. The CNN was designed and trained to use the geometry, boundary conditions, and load as input to predict the stress contours. The proposed technique's performance was compared to Finite-Element simulations using a partial differential equation (PDE) solver.

Chapter III is focused on integrating physics knowledge into a convolutional neural network (CNN) to boost learning within a feasible solution space in a specific domain. Our proposed method uses deep neural networks in the form of (CNNs) augmented with custom loss functions that use physics rules to bypass the need for Finite Element Analysis and predict high-resolution stress distributions on damaged steel plates with variable loading and boundary conditions. We embedded physics constraints into the loss function to enforce the model training, precisely capturing stress concentrations around the tips of various structural damage configurations. The CNN was designed and trained to use the geometry, boundary conditions, and load as input and predict the stress contours. The proposed framework's performance is compared to Finite-Element simulations using partial differential equation (PDE) solver.

Chapter IV outlines the development of a deep learning model, Neuro-DynaStress, which is proposed to predict the entire sequence of dynamic stress distribution based on finite element simulations using a partial differential equation (PDE) solver. The model was designed and trained to use the geometry, boundary conditions, and sequence of loads as input and predict the sequences of high-resolution stress contours. The proposed framework's performance is compared to finite

element simulations using a PDE solver. The goal is to train a model that can later be used when real-time estimations are needed, such as in the aftermath of extreme disruptive events.

Chapter V presents a deep learning model to reduce computational cost while maintaining accuracy, a Physics Informed Neural Network (PINN), the 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.

Finally, Chapter VI summarizes the work performed under this project, outlines the main research products developed and presents the main findings of the study. Some directions for future research are also presented.

CHAPTER 2

HIGH-RESOLUTION STATIC STRESS DISTRIBUTION PREDICTION IN INTACT STRUCTURAL COMPONENTS

2.1 Methodology

Deep Learning (DL) techniques can generate results significantly faster than conventional run-time analysis. This can prove extremely valuable in real-time structural assessment applications. Our proposed method uses deep neural networks in the form of convolutional neural networks (CNN) to bypass the FEA and predict high-resolution static stress distributions on loaded steel plates with variable loading and boundary conditions. The CNN was designed and trained to use the geometry, boundary conditions, and load as input to predict the stress contours. An overview of our approach is shown in Fig. 2.1.

2.1.1 Data generation

Two-dimensional steel plate structures with five edges, E1 to E5 denoting edges 1 to 5, as shown in Fig. 2.2, are considered to be made of homogeneous and isotropic linear elastic material. The 2D steel plates have similar geometry to that of gusset plates, as used for connecting beams and columns to braces in steel structures. The boundary conditions and loading angles simulate conditions that are similar to those affecting common gusset plate structures under external loading. Analysis of the behavior of these components is essential since various reports have observed failures of gusset plates subject to lateral loads [41, 42, 43, 44]. The distributed static loads applied to the plates in this study range from 1 to 5 kN with intervals of 1 kN. Moreover, loads are applied with three angles $\pi/6$, $\pi/4$, and $\pi/3$, on either one or two edges of the plate. The load is decomposed to its horizontal and vertical direction components. The boundary conditions and load cases are considered to simulate similar conditions in common gusset plate structures under external loading. Some of the most common gusset plates configurations in practice are shown in Fig. 2.3. Four boundary conditions are considered, as shown in Fig. 2.4, based on real gusset plates' boundary

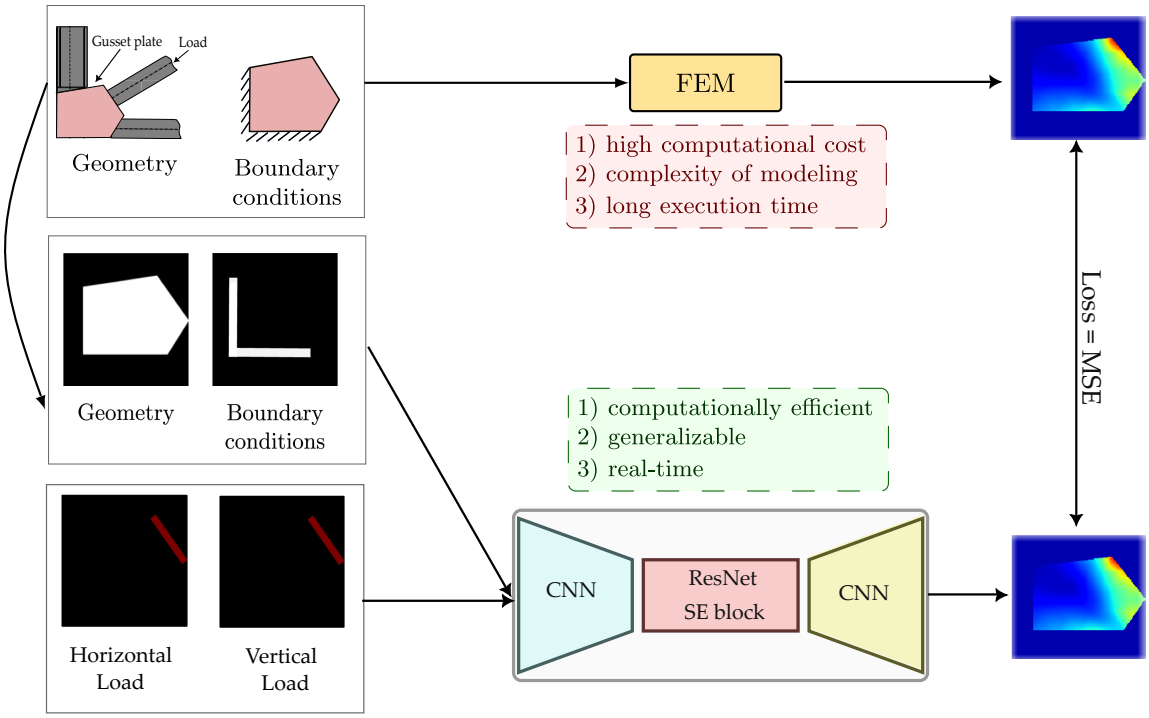


Figure 2.1 **Overview:** Unlike FEM, our proposed model is computationally efficient and facilitates real-time analysis. The existing workflow for FEM applications includes: (i) modeling the geometry and its components, (ii) specifying material properties, boundary conditions, meshing, and loading, (iii) analysis, which may be time-consuming based on the complexity of the model. Our model takes geometry, boundary condition, and load as input and predicts the static stress distribution.

conditions.

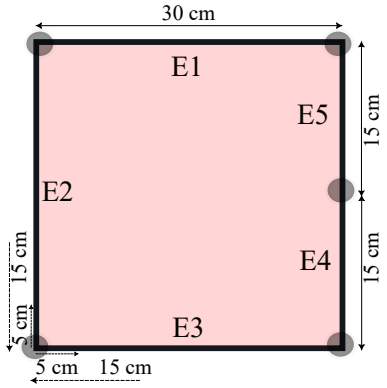


Figure 2.2 Basic schematic topology for initializing the steel plate geometries.

All the translational and rotational displacements are fixed at the boundary conditions. All input variables used to initialize the population are shown in Table 1. The minimum and maximum range for the width and height of the plate are from 30 to 60 cm. Various geometries are generated by changing the position of each node in horizontal and vertical directions, as shown in Fig. 2.2,

which lead to 1024 unique pentagons. The material properties remain unchanged and isotropic for all samples.

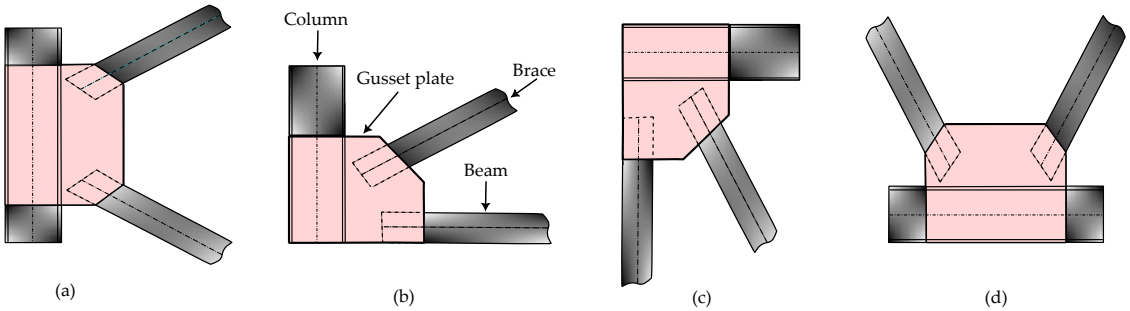


Figure 2.3 Some of the most common gusset plates in practice.

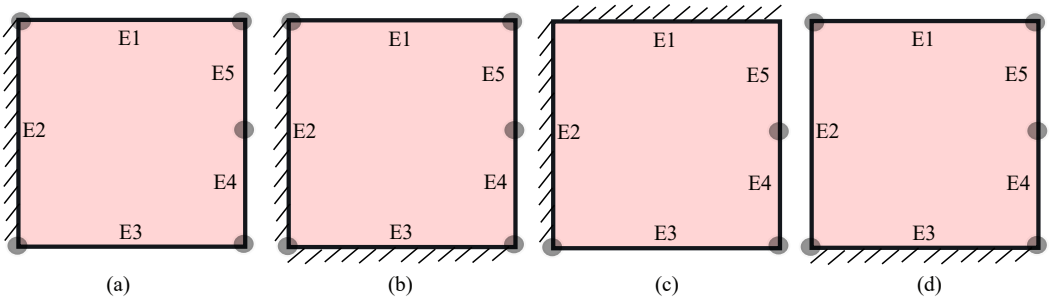
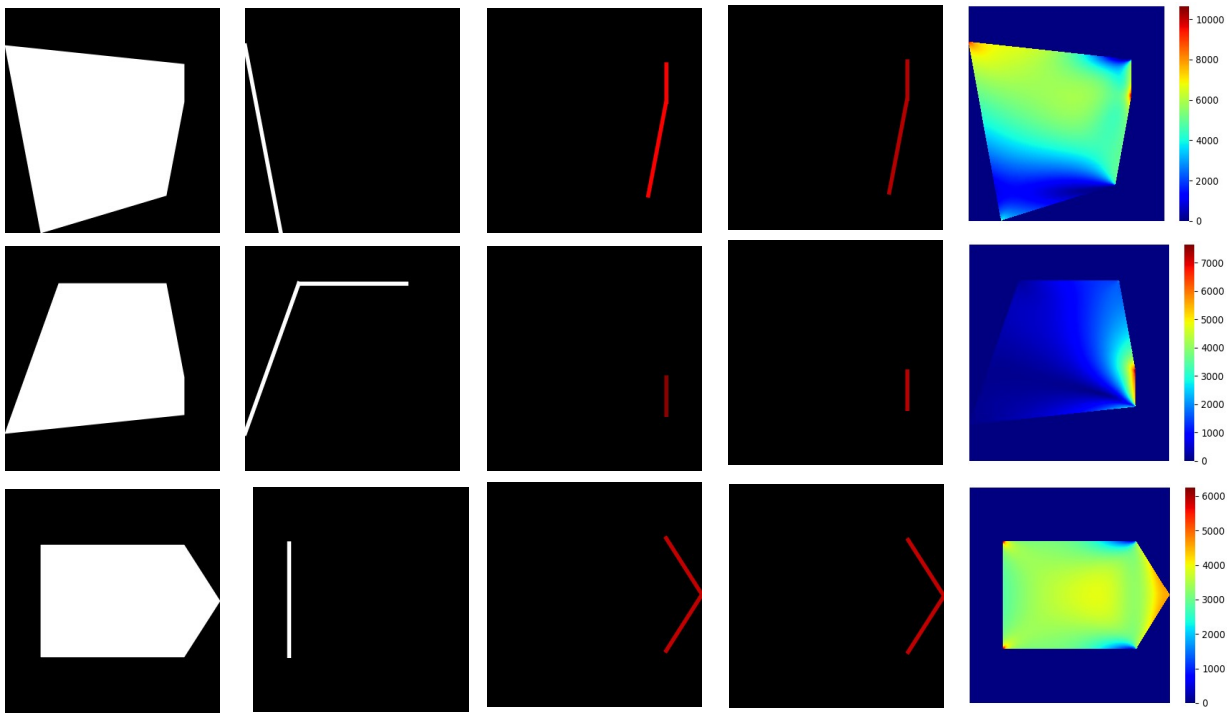


Figure 2.4 Different types of boundary conditions for initializing population.

2.1.1.1 Input Data

The geometry is encoded into a 600 × 600 matrix as a single channel binary image. 0 (black) and 1 (white) denote the outside and inside of the geometry, as shown in Fig. 2.5(a). The boundary condition is also represented by another 600 × 600 pixel binary image, where the constrained edges are defined by 1 (white) (Fig. 2.5 (b)). Moreover, each horizontal and vertical component of the load is encoded as one 600 × 600-pixel single-channel colored image, as shown in Fig. 2.5(c) and 2.5(d). Each row of Fig. 2.5 represents one of the simulated geometry, boundary conditions, and load positions as described in table 2.1. The magnitude of the horizontal and vertical components of the loads, after decomposition, varies between 0.5 kN and 4.33 kN. These loads are normalized between (100,0,0) and (255,0,0) as RGB colors to create a color image where the colored part represents the location and magnitude of the load (Figs. 2.5(c) and 2.5(d)).



(a) (b) (c) (d) (e)

Figure 2.5 Input and output representation for static stress distribution prediction: (a) geometry, (b) boundary condition, (c) horizontal load, (d) vertical load, (e) output.

Table 2.1 Input variable

Geometry	Boundary conditions	Load position	Load angle(degree)	Load magnitude (kN)
pentagon	E2	E5,E4,E4E5	30,45,60	1,2,3,4,5
pentagon	E2E3	E5	30,45,60	1,2,3,4,5
pentagon	E1E2	E4	30,45,60	1,2,3,4,5
pentagon	E3	E2E5,E1E2E5	30,45,60,90	1,2,3,4

2.1.1.2 Output Data

FEA is performed using the Partial Differential Equation (PDE) solver in the MATLAB toolbox to obtain the stress distributions of each sample. The MATLAB PDE toolbox mesh generator only generates unstructured triangulated meshes incompatible with CNN. Since each element should be represented by one pixel in an image, we develop a 600 × 600 grid surface equal to the dimensions of the most significant possible geometry. The stress values are then interpolated between the triangular elements and grids to determine a stress distribution compatible with our CNN network. The stress values of all the elements outside the material geometry are assigned a zero, as shown

in Fig. 2.5(e). The dimension of the largest sample is 600 mm × 600 mm, and the smallest is 300 mm × 300 mm. Therefore, the size of each element is 1 mm × 1 mm, which means that each image has 360000 pixels. This high-resolution dataset offers significant accuracy. The maximum and minimum von Mises stress values for elements among the entire dataset are 96,366 MPa and –0.73 MPa, respectively. We normalized all the output data between 0 and 1 to ensure faster convergence and encoded it to 600 × 600 matrices.

2.1.2 Model Architecture

The CNN can be built using a sequence of convolutional layers. The convolutional layers learn to encode the input in simple signals and reconstruct the input [45]. We conducted an ablation study to determine the best network architecture for our convolutional and deconvolutional layers, as well as a number of Squeeze-Excitation and Residual blocks. We tested a range of different architectures, varying the number of convolutional and deconvolutional layers, and the number of Squeeze-Excitation and Residual blocks. Through our tests, we were able to identify the most accurate network architecture for our needs. As shown in table 2.2 arch 1 has minimum relative error compared to its counterparts. This network architecture was then used for the rest of our experiments. This architecture has three stages of layers: The first stage is downsampling which consists of seven convolutional layers (E1, E2, E3, E4, E5, E6, E7), and the second stage has three layers (RS1, RS2, and RS3) of Squeeze-Excitation and Residual blocks (SE-ResNet). In addition, the Inception and MobileNetV2 blocks are swapped with the SE-ResNet block to check if these modules can further enhance the network's performance. The third stage is upsampling, consisting of six deconvolutional layers (D1, D2, D3, D4, D5, D6), as illustrated in Fig. 2.6.

Table 2.2 Choice of architecture

Item	Architecture								
	arch 1	arch 2	arch 3	arch 4	arch 5	arch 6	arch 7	arch 8	arch 9
Conv layers	7	6	8	7	6	8	7	6	8
SE-ResNet layers	3	3	3	2	2	2	4	4	4
ConvT layers	6	5	7	6	5	7	6	5	7
PMAE (%)	0.9	1.02	1.21	1.13	1.35	1.49	1.08	1.27	1.40

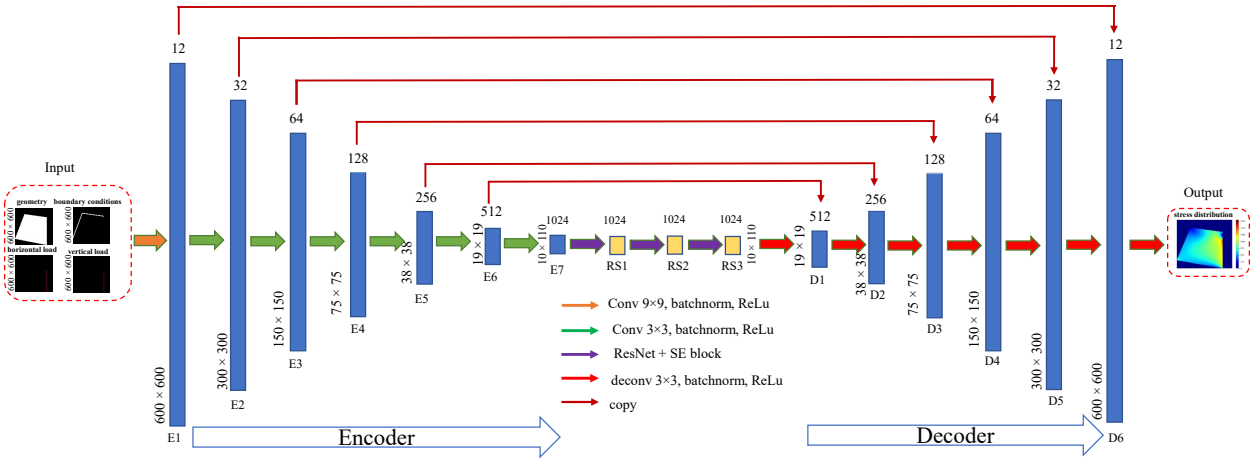


Figure 2.6 Proposed CNN architecture.

2.1.2.1 Residual Block

We used residual blocks to address the vanishing gradient problem. In addition, SE blocks are computationally lightweight and result in only very small increases in model complexity. As illustrated in Fig. 2.7, the formulation of F(x)+x can be realized by feedforward neural networks with shortcut connections. The shortcut connection simply performs identity mapping, and its output is added to the output of the stacked layers [46].

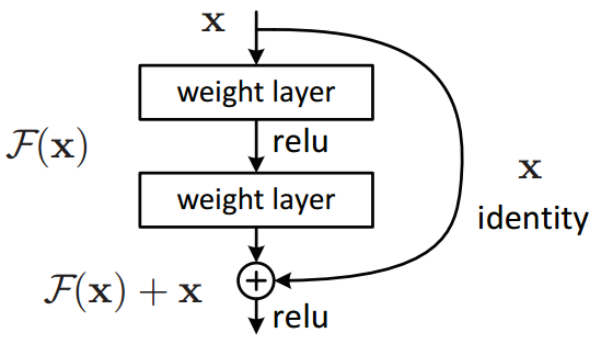


Figure 2.7 The building block of residual learning [46].

2.1.2.2 Squeeze-and-Excitation Blocks

As depicted in Fig. 2.8, Squeeze-and-Excitation blocks improve the representational capacity of the network, enabling dynamic channel-wise feature recalibration. A SE-block can be implemented with five steps. First, we feed the input x as a convolutional block and the current number of channels to the SE function, where F_{tr} in Fig. 2.8 is the convolutional operator for the transformation of X to U. Then, at the second phase, Each channel is squeezed into a single numeric value by using average

pooling. Additionally, in the third phase, a fully connected layer is followed by a ReLu function, which applies a nonlinearity and reduces the output channel complexity. Then in the fourth phase, SE blocks can be used directly with residual networks. Fig. 2.9 depicts a SE-ResNet module in the SE block transformation. F_{tr} is regarded as the non-identity branch of a residual module. Before the summation of the identity branch, both squeeze and excitation acts. Using both SE and ResNet in the network outperforms using ResNet [47].

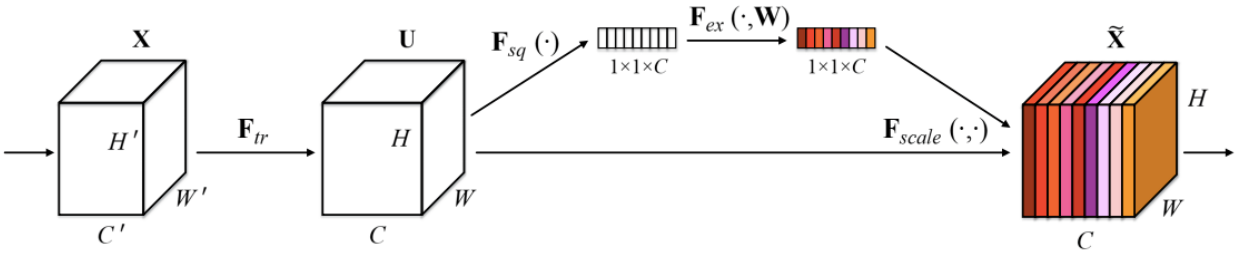


Figure 2.8 The building block of Squeeze-and-Excitation [47].

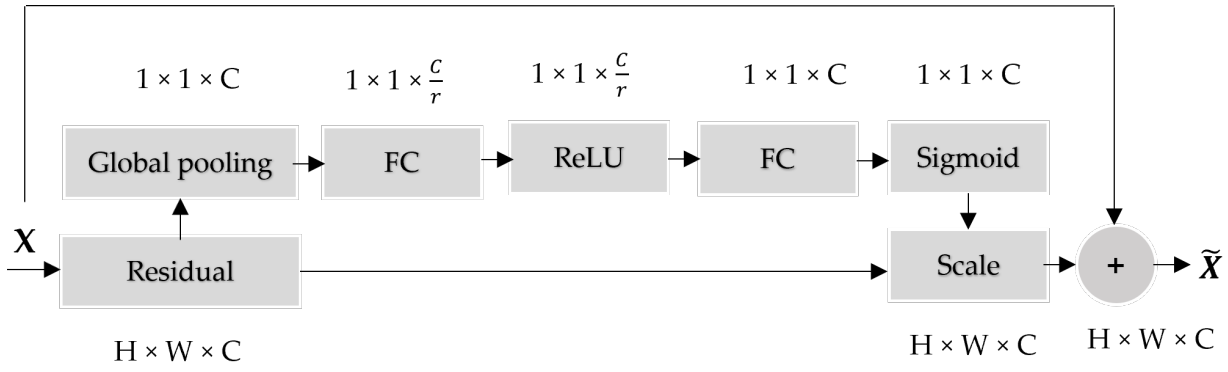


Figure 2.9 SE-ResNet module.

2.1.2.3 Inception Block

Inception Modules are used to reduce the computational cost of CNNs. Since neural networks have to deal with a vast array of images, each with different content, they must be carefully designed. Using the vanilla version of the inception module, we can perform a convolution on the input meaning three different sizes of filters (1 × 1, 3 × 3, 5 × 5) instead of one. Also, max pooling is performed. The outputs are then concatenated and sent to the next layer. Therefore, convolutions occur at the same level in CNNs, where the network gets wider, not deeper. Compared with shallower

and less wide CNNs, this method offers significant quality gains at a modest computational cost increase [48]. Fig. 2.10 depicts the inception module.

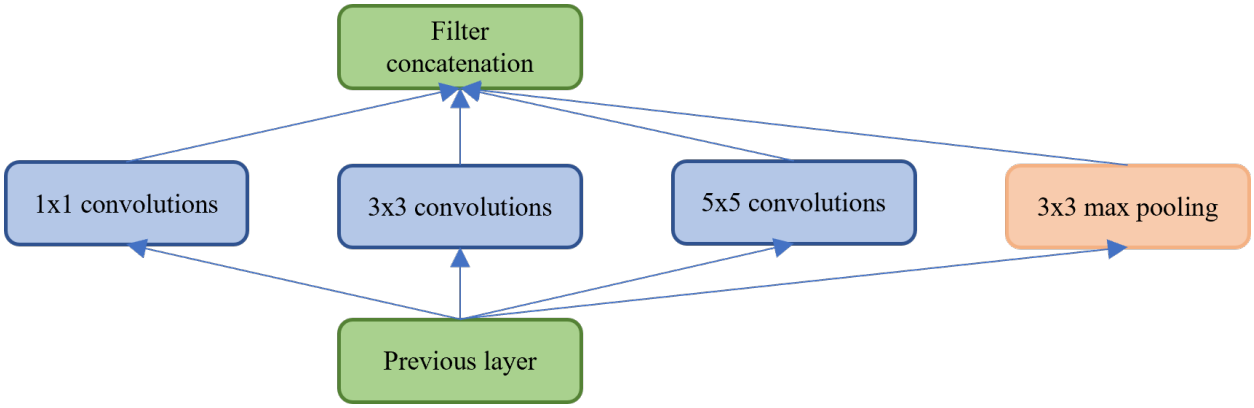


Figure 2.10 Inception module.

2.1.2.4 MobileNetV2 Block

MobileNetV2 is based on an inverted residual block with shortcut connections between thin bottleneck layers [49]. A lightweight depth-wise convolution technique is used in the intermediate layer to filter features as a source of nonlinearity. The nonlinearities must be removed in the narrow layers to maintain representational power. In general, in this model, the bottlenecks encode the intermediate inputs and outputs of the model, while the inner layer encodes how the model can transform from lower-level concepts such as pixels to higher-level features such as categories of images. Lastly, shortcuts can improve training speed and accuracy, just like traditional residual connections. Fig. 2.11 depicts the MobileNetV2.

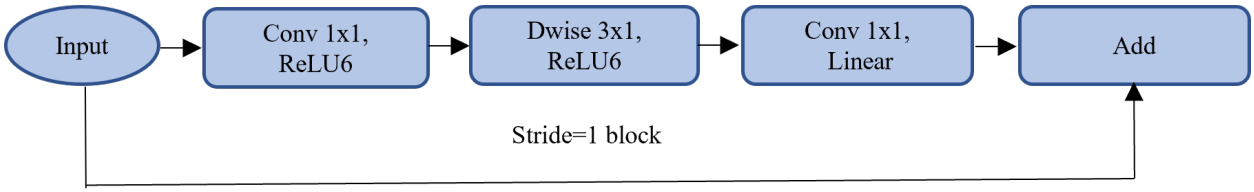


Figure 2.11 MobileNetV2 module.

2.1.2.5 Network layers and hyperparameters

All the details of the network layers and hyperparameters can be found in tables 2.3 and 2.4. As can be seen, the models consist of seven Conv layers, three different bottleneck blocks, and six ConvT layers. Since arch 1 shows the best performance among all architectures; therefore, we

keep the network with 1024 channels as the primary model and swap the bottleneck each time with the SE-ResNet, Inception, and MobileNetV2. We keep the bottleneck dimension the same for all models to match the ConvT first layer. The batch size is set to 16, leading to the best accuracy compared to other batch sizes. Different learning rates from 1e−3 to 1e−6, and 1e−5 lead to the best convergence.

Table 2.3 Network layers

item	Number of layers	First layer(H×W×C)	Last layer(H×W×C)	Activation
Conv	7	600×600×12	10×10×1024	ReLU
SE-ResNet	3	10×10×1024	10×10×1024	Sigmoid-RelU
Inception	3	10×10×1024	10×10×1024	ReLU
MobileNetV2	1	10×10×1024	10×10×1024	ReLU6
ConvT	6	19×19×512	600×600×12	ReLU

Table 2.4 Network hyperparameters

batch size	Learning rate	Weight decay	Expand ratio	Loss function
16	1e−5	1e−7	6	MSE-MSE

2.2 Loss function and performance metrics

We used MSE (mean squared error) for the training loss defined in Eq. 2.1. MSE gives a more significant penalty to large errors than MAE (mean absolute error), and the errors, also are normally distributed. Using MAE (mean absolute error), MRPE (mean relative percentage error), PMAE (percentage mean absolute error), PAE (peak absolute error) and PPAE (peak percentage absolute error) helps evaluate the overall quality of predicted stress distribution. These metrics are defined in Equations 2.2 to 2.6, respectively.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(S(i) - S^{\wedge}(i))^2 \tag{2.1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|(S(i) - S^{\wedge}(i)| \tag{2.2}$$

where $S(i)$ is the stress value at a node 'i' computed by FEA as the ground truth, $S^{\wedge}(i)$ is the corresponding predicted stress by the DL model, and n is the total number of elements at each sample which is 360000 in our work. Symbol || denotes the absolute value. Our model's prediction and ground truth are displayed as 600×600 resolution images.

We used MRPE for measurement of the relative error:

$$MRPE = \frac{MAE}{\max(|S(i)|, |\hat{S(i)}|)} \times 100 \tag{2.3}$$

The percentage mean absolute error is defined as:

$$PMAE = \frac{MAE}{max[S(i)] - min[S(i)]} \times 100 \tag{2.4}$$

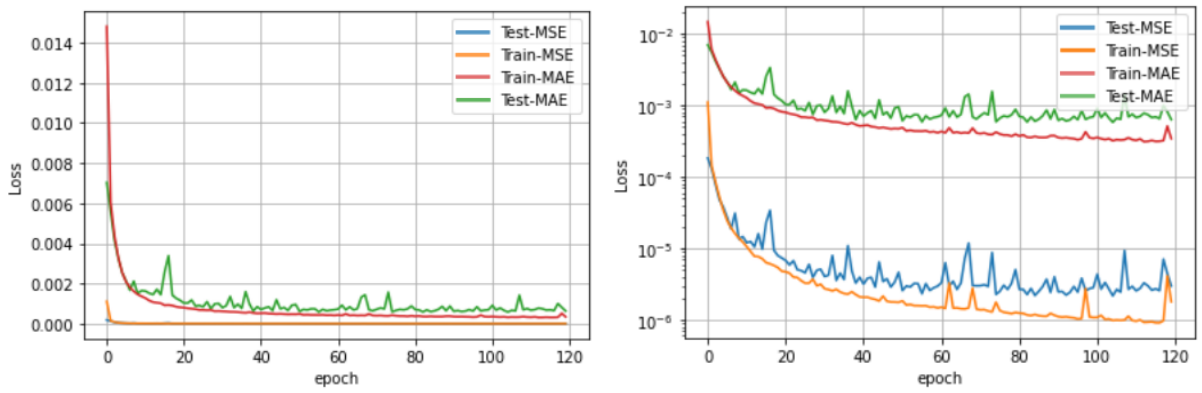
where max $[S(i)]$ is the maximum value in a set of ground truth stress values and min $[S(i)]$ is the minimum value. PAE and PPAE measure the accuracy of maximum stress which are one of the main important critical load values in the predicted stress distribution. The importance of maximum stress matters in the design phase since maximum stress should be less than yield strength to avoid permanent deformation. PAE and PPAE are defined as:

$$PAE = [S(i)] - [S^{\wedge}(i)] \tag{2.5}$$

$$PPAE = \frac{PAE}{[S(i)]} \times 100 \tag{2.6}$$

2.3 Results and discussion

All codes are written in PyTorch Lightning and run on two NVIDIA TITAN RTX 24G GPUs. We use the AdamW (Adam algorithm with weight decay) optimizer to speed up the convergence



(a) (b)

Figure 2.12 MSE and MAE curves on training and testing data with two scales: (a) linear scale, (b) logarithmic scale.

of models. We train and evaluate different models based on Table 3 to find the model with the best performance. The training data size of models 1 to 3 is 83,558, and the testing data size is 20,890, randomly divided with a train/test ratio of 80%–20%. Fig. 2.12 shows MSE and MAE losses as a function of epochs in model 1. Figs. 2.12a and Fig. 2.12b are linear and logarithmic scales. Fig. 2.12a shows that the MSE and MAE curves rapidly decline after a few epochs. However, Fig. 2.12b gives a more precise representation of the model's behavior. Fig. 2.12b shows that MSE is smaller than MAE, but both have similar general trends.

We save the best checkpoint during validation, and all error metrics are based on the best checkpoint. Models 4 to 6 are validated with K-fold cross-validation to ensure that the model is generalizable. To reduce the computational cost, we divide the dataset into three folds. K-fold cross-validation shows the best performance in all models based on most metrics, as can be seen in Table 2.5, which means the model is generalizable.

We replace the SE-ResNet block in the bottleneck with the Inception and MobileNetV2 block in models 2 and 3, respectively. Model 1, has the best performance in terms of PPAE, with an error of 0.46% and model 2 is the best model, based on PMAE with a 0.57% error. Fig. 2.13 depicts the performance of different models in terms of MAE. As can be seen in Fig. 2.13, models 3 and 1, which have MobileNetworkV2 and SE-ResNets in a bottleneck, have almost the same performance, and model 2 with inception block is the best in terms of MAEs. We deem these

Table 2.5 Error metrics for models at best checkpoints (Units:MPa)

model	dataset	bottleneck	MSE	MAE	MRPE(%)	PMAE(%)	PPAE(%)	PAE
Model 1	classic	SE-ResNet	0.21	58.8	3.80	0.9	0.46	30.00
Model 2	classic	Inception	0.62	31.5	1.54	0.57	2.66	150.55
Model 3	classic	MobileNet	0.38	51.99	2.83	0.93	4.36	246.50
Model 4	Fold 1	SE-ResNet	0.59	34.90	1.80	0.63	2.19	124.00
Model 5	Fold 2	SE-ResNet	0.64	25.04	1.10	0.45	0.12	6.93
Model 6	Fold 3	SE-ResNet	0.61	32.03	1.42	0.57	0.93	52.89
Mean			0.61	30.65	1.44	0.55	1.08	61.27
STD			0.02	5.14	0.28	0.07	0.85	58.15

results satisfactory for stress distribution predictions, specifically the PPAE, the most critical load value for stress distribution in engineering domain applications.

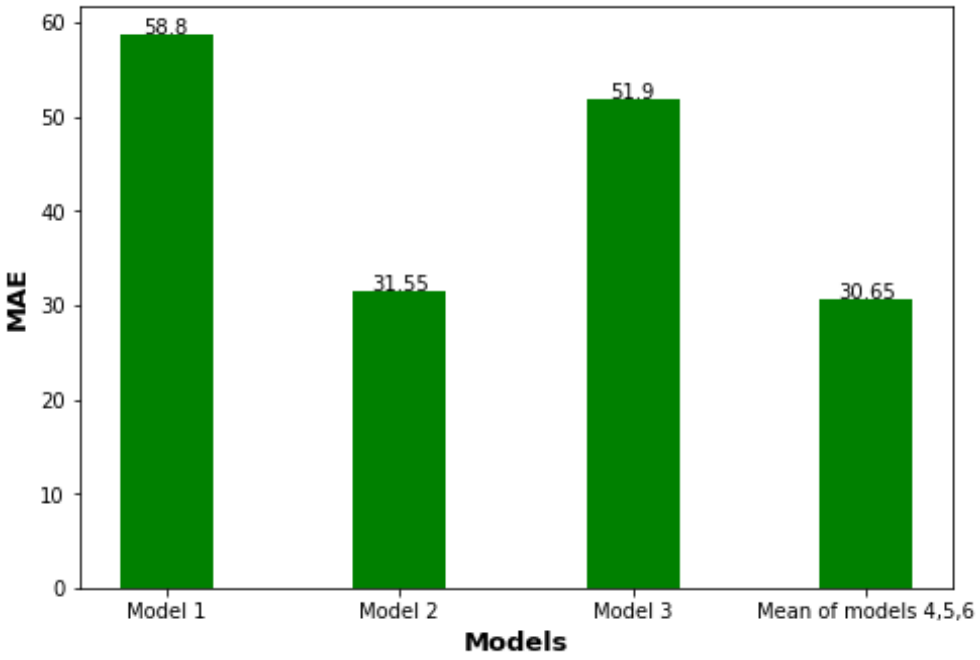


Figure 2.13 Comparison of different models in terms of MAE.

The predictions produced from some randomly selected samples from the test dataset of model 1 are visualized in Fig. 2.14 Each row represents a sample. Columns (a) to (d) represent geometry, boundary conditions, and load in horizontal and vertical directions, respectively. Columns (e) and (f) represent the ground truth and predicted stress distributions. As can be seen, there is a high fidelity fit between ground truth and predicted stress distributions in both maximum stress and stress distribution in the different samples. Also, some inaccurate predictions are shown in Fig. 2.15.

These predictions still provide information. Fig. 2.16 illustrates the cumulative distribution of PMAE and PPAE in the test dataset of model 1. Fig. 2.16a shows the probability of mean in PMAE is 80%, which means that about 80% of predicted samples have a PMAE of less than 0.9, and 50% of samples have a PMAE of less than 0.46, which is the median. Fig. 2.16b shows that about 99% of predicted samples have a PPAE of less than 0.46, and 50% of the predicted samples have a PPAE of 0.06.

2.3.1 Effect of dataset size on the performance of the network

We break the data into different sizes to evaluate the effect of data size on the network's performance of model 1. Therefore, besides training with the entire dataset, 104448 samples, we train the network with 10,000, 20,000, 30,000, 40,000, 50,000, and 70,000 samples. Fig. 2.17 demonstrates that training with just 10% of the dataset can achieve a mean error of 1.85%, which is acceptable in most engineering applications. Also, it can be seen that if we want to accomplish a mean error of less than 1%, we should train the network with at least 90% of the dataset. We also evaluate the effect of data size on the Gaussian distributions of PMAE and PPAE, illustrated in Figs. 2.18a and 2.18b. As shown in Fig. 2.18a, increasing the data size decreases the standard deviation of PMAE. However, a 70,000 data size and the total data size have almost the same standard deviation. Fig. 2.18b shows that the standard deviation of PPAE decreases when the data size increases from 50000 to 70000. As a result, we should train the network with at least 70,000 examples, 67% of our dataset, to ensure PPAE's acceptable standard deviation accuracy.

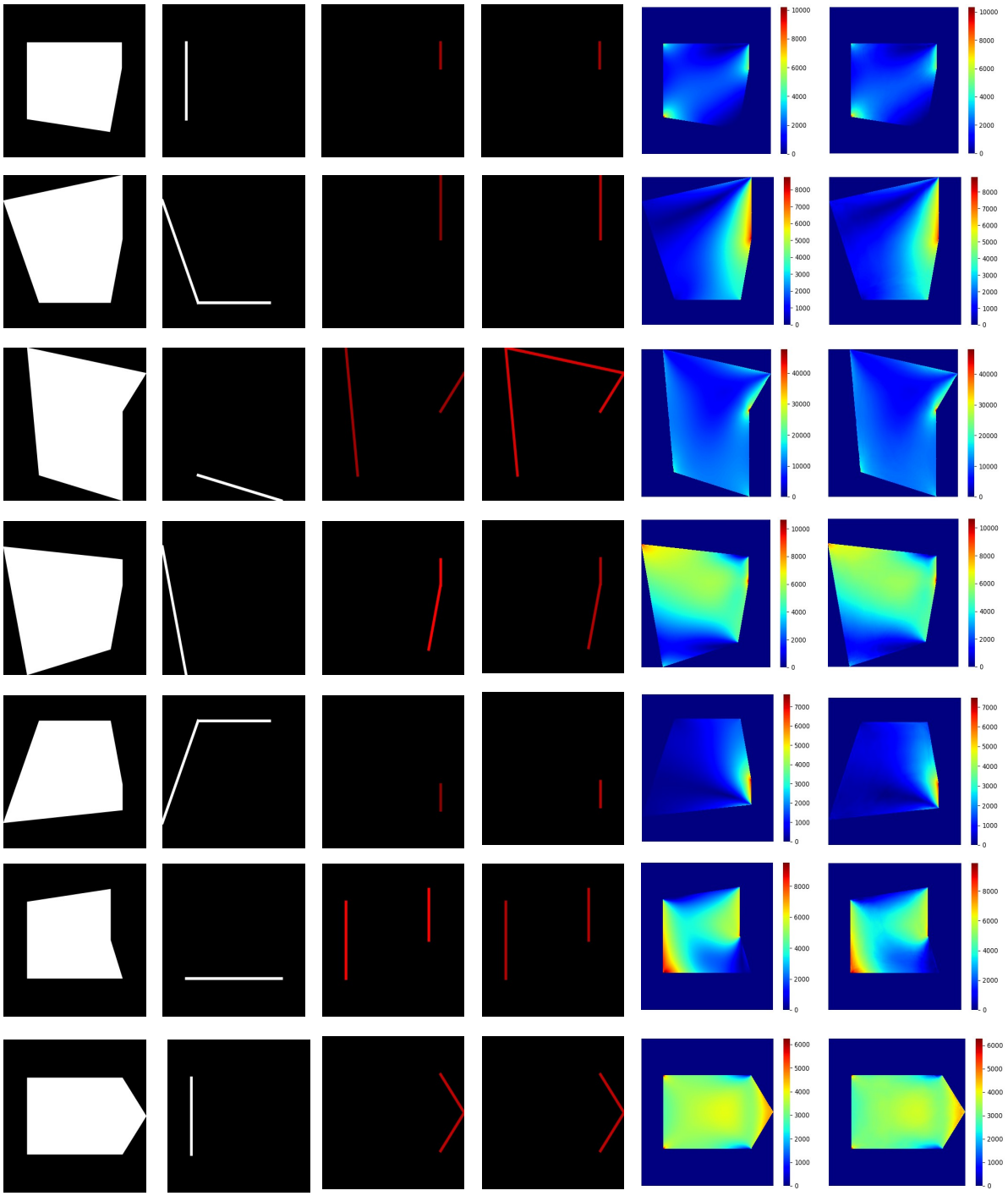
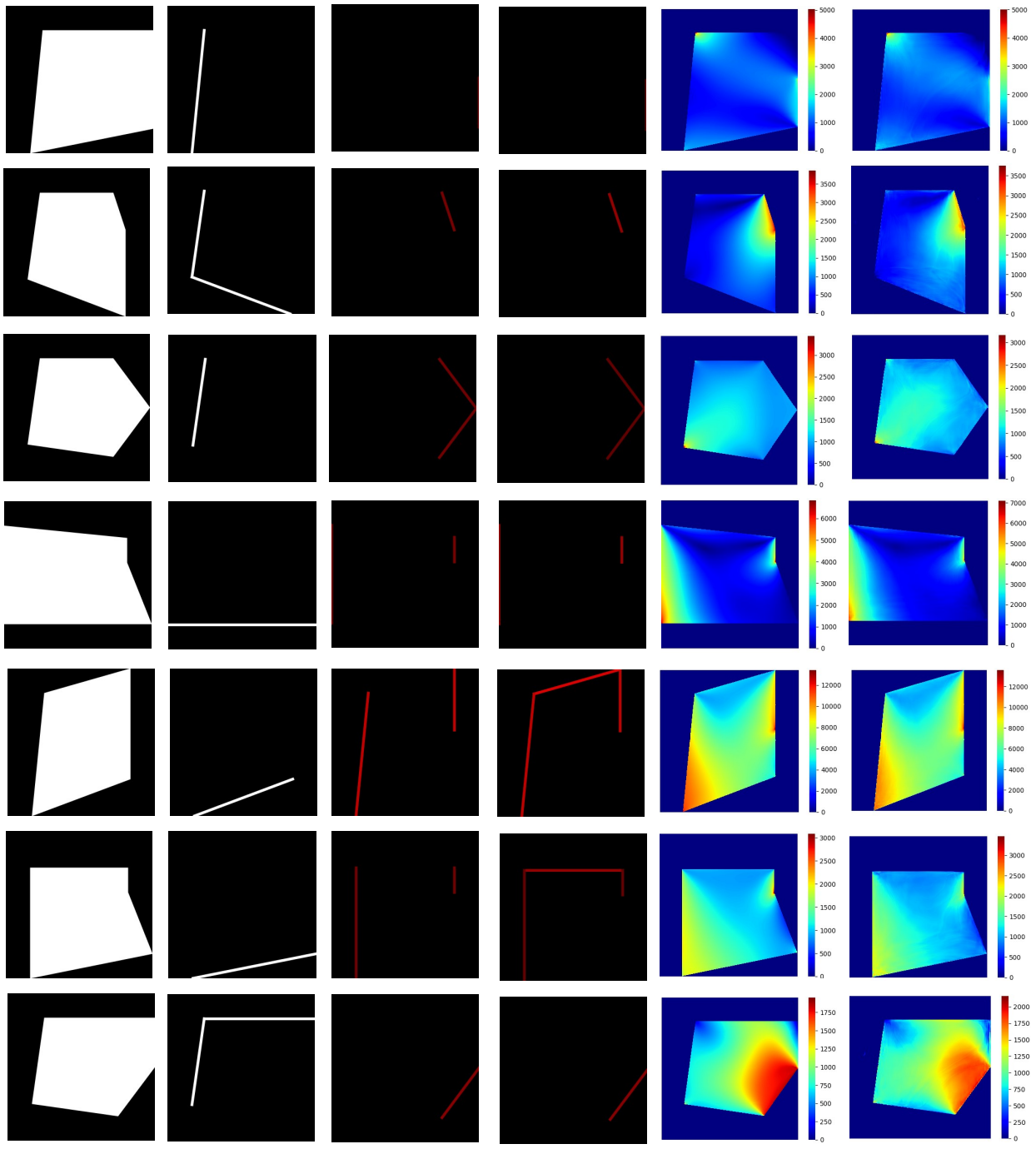
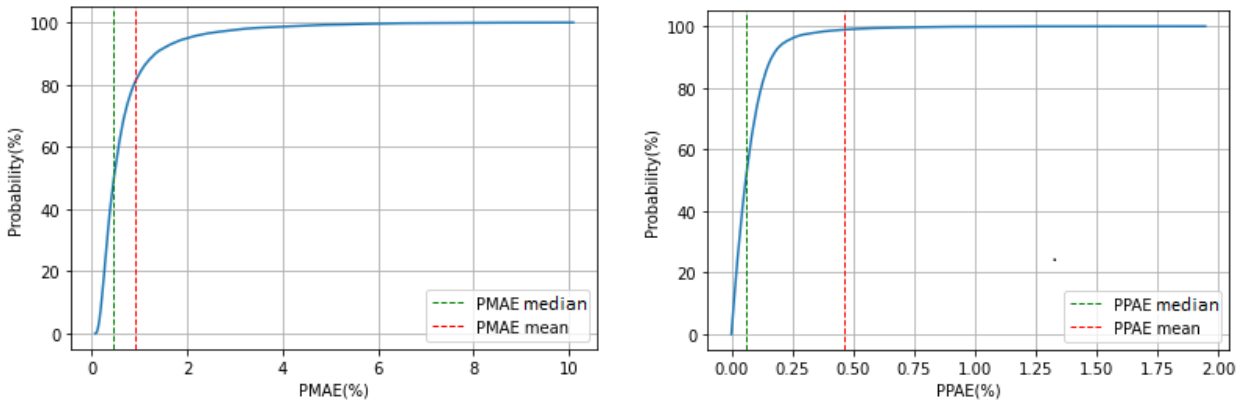


Figure 2.14 Predicted stress distribution and corresponding inputs with different loads and boundary conditions scenarios. Columns (a) to (d) represent (a) geometry, (b) boundary conditions, and load in a (c) horizontal and (d) vertical direction. Column (e) represent ground truth, and column (f) shows predicted stress distribution, respectively (Units = MPa).



(a) (b) (c) (d) (e) (f)

Figure 2.15 Inaccurate predicted stress distribution and corresponding inputs with different loads and boundary conditions scenarios. Columns (a) to (d) represent (a) geometry, (b) boundary conditions, and load in a (c) horizontal and (d) vertical direction. Column (e) represent ground truth, and column (f) shows predicted stress distribution, respectively (Units = MPa).



(a) (b)

Figure 2.16 Cumulative distribution of PMAE and PPAE: (a) PMAE of samples less than mean and median on the test dataset, (b) PPAE of samples less than mean and median on the test dataset.

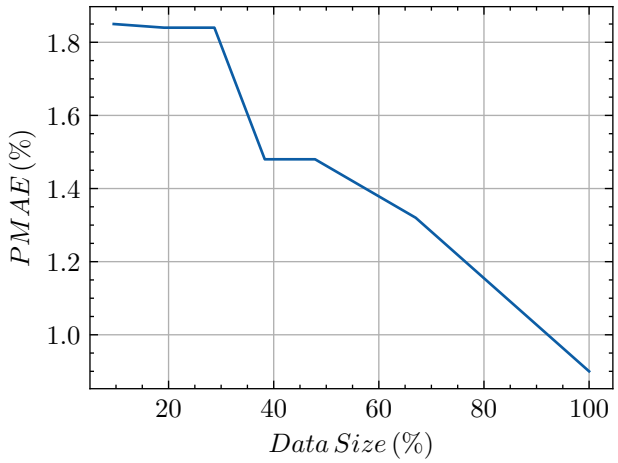
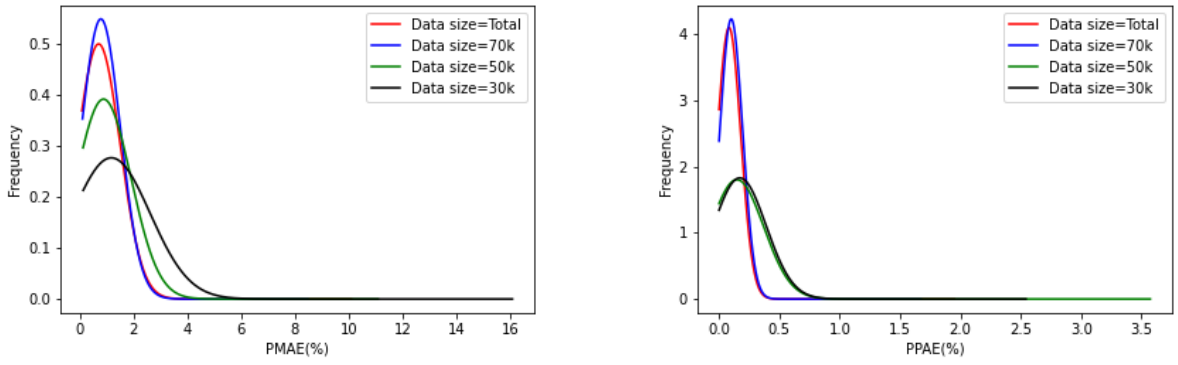


Figure 2.17 PMAE at different data sizes.



(a) (b)

Figure 2.18 Gaussian distribution of PMAE and PPAE. (a) PMAE, (b) PPAE.

CHAPTER 3

HIGH-RESOLUTION STATIC STRESS DISTRIBUTION PREDICTION IN DAMAGED STRUCTURAL COMPONENTS

3.1 Predicting physical responses

Accurately and efficiently predicting physical responses is essential for different real-world applications, such as the prediction of the remaining life of mechanical systems [50], earthquake alarms [51], and weather forecasting [52]. Although data-driven and physics-based solutions allow for solid predictions, both methods still suffer from several limitations. The drawbacks of data-driven methods are the requirement for large amounts of data, the inability to produce physically consistent results, and the lack of generalization to out-of-sample scenarios. [53]. However, physics-based models, such as Finite Element Analysis (FEA), are computationally prohibitive. Therefore, to achieve fast analysis of mechanical systems and address deficiencies of data-driven models, we integrate conventional physics-based methods with state-of-the-art Deep Learning (DL) methods to predict stress distributions in damaged steel components.

3.2 Methodology

3.2.1 Data generation

Two-dimensional steel plate structures with five edges, E1 to E5 denoting edges 1 to 5 as shown in Fig. 3.2, are considered homogeneous and isotropic linear elastic material. Various geometries are generated by changing the position of each node in horizontal and vertical directions, as shown in Fig. 3.2, which led to 1024 unique pentagons. The material properties remain unchanged and isotropic for all samples. 1024 crack scenarios with various widths, lengths, angles, and locations were also created on the steel plates. Since the number of geometries is the same as the number of crack scenarios, each geometry has a unique crack.

The 2D steel plates approach the geometry of gusset plates. The boundary conditions and

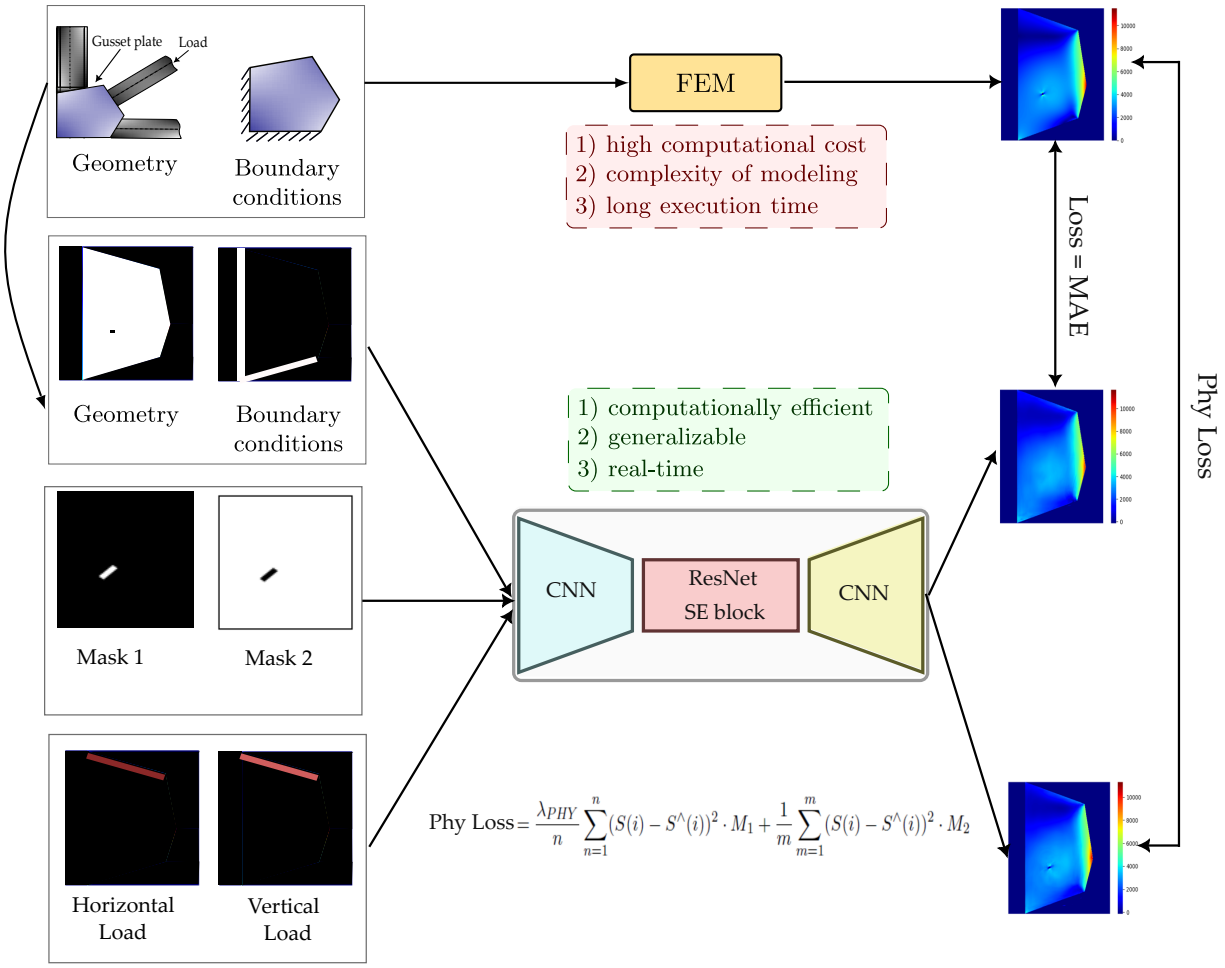


Figure 3.1 **Overview:** Phy loss facilitates localizing the cracks and capturing stress concentrations around the crack tips.

loading angles are considered to simulate similar conditions in common gusset plate structures under external loading. Adding different loading and boundary conditions extended the population into 61,440 unique samples. All input variables used to initialize the population are shown in Table 3.1. Gusset plates are used for connecting beams and columns to braces in steel structures.

Table 3.1 Input variable

Geometry	Boundary conditions	Load position	Load angle(degree)	Load magnitude (kN)
pentagon	E2	E4E5	30,45,60	1,2,3,4,5
pentagon	E2E3	E5	30,45,60	1,2,3,4,5
pentagon	E1E2	E4	30,45,60	1,2,3,4,5
pentagon	E1	E2	30,45,60	1,2,3,4,5

For crack initiation, we divided the steel plate into 15 different regions to create cracks with

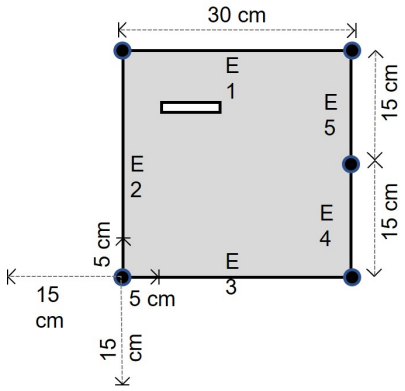


Figure 3.2 Basic schematic topology for initializing the damaged steel plate geometries.

different lengths and ensure that the crack length would not violate the edges of the steel plate. Fig. 3.3 shows all damage locations in 3 categories. Every red point represents the center of single damage in the steel plate. Categories 1 to 3 each have 9, 4, and 2 subcategories, respectively. Each plate has just one crack, and the other red points just represent the location of the cracks. Details of crack initiation are shown in Table 3.2.

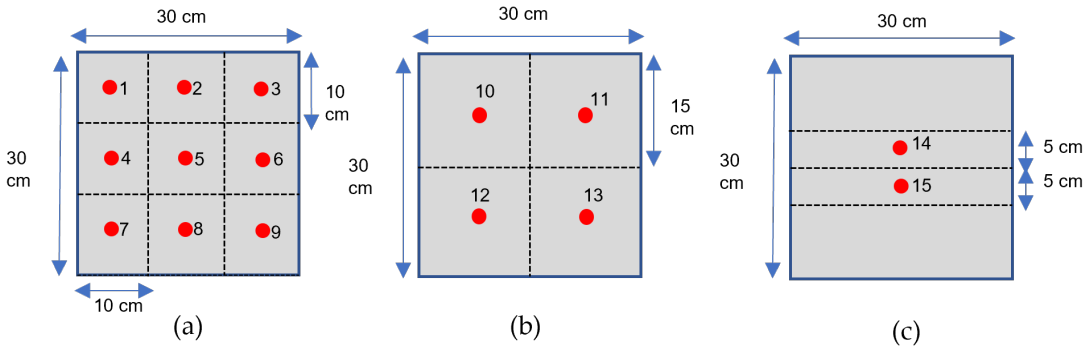


Figure 3.3 Different location of damages on steel plates: (a) category 1, (b) category 2, (c) category 3.

The distributed static loads applied to the gusset plates in this study ranged from 1 to 5 kN with intervals of 1 kN. Moreover, loads were applied with 3 different angles, $\frac{\pi}{6}$, $\frac{\pi}{4}$ and $\frac{\pi}{3}$ on either one or two edges of the plate. The load is decomposed into its horizontal and vertical direction components. Also, four types of boundary conditions are considered, as shown in Fig. 3.4, similar to real gusset plates' boundary conditions. All the translational and rotational displacements were fixed at the boundary conditions. The minimum and maximum range for the width and height of the plates are from 30 cm to 60 cm.

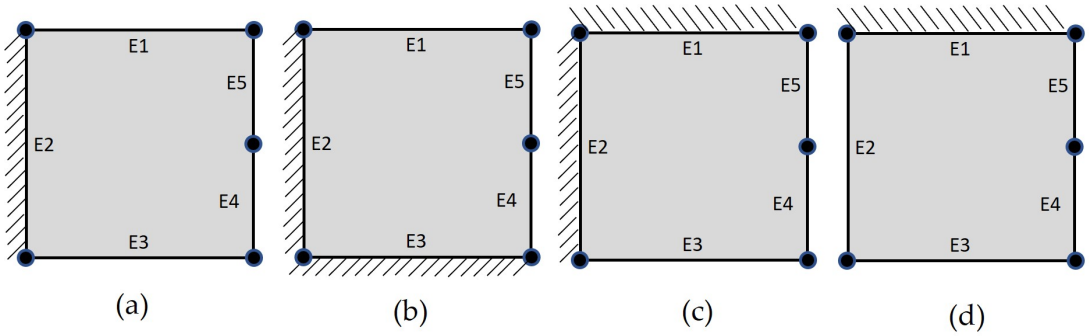


Figure 3.4 Different types of boundary conditions for initializing population.

Table 3.2 Detail of damages in steel plates

Crack number	Width (mm)	Length of category1 (mm)	Length of category2 (mm)	Length of category3 (mm)	Angle (degree)
1	1	10,40,80	120	160,200	0,30,45,90
2	2	10,40,80	120	160,200	0,30,45,90
3	3	10,40,80	120	160,200	0,30,45,90
4	4	10,40,80	120	160,200	0,30,45,90
5	5	10,40,80	120	160,200	0,30,45,90
6	6	10,40,80	120	160,200	0,30,45,90
7	7	10,40,80	120	160,200	0,30,45,90

3.2.1.1 Input Data

The geometry is encoded into a 600 × 600 matrix as a single channel binary image. 0 (black) and 1 (white) denote the outside and inside of the geometry, as shown in Fig. 3.5a. The boundary condition is also represented by another 600 × 600 pixel binary image, where the constrained edges are defined by 1 (white) as shown in Fig. 3.5b. Moreover, each horizontal and vertical component of the load is encoded as one 600 × 600-pixel single-channel colored image, as shown in Figs. 3.5c and 3.5d. Each row of Fig. 3.5 represents one of the simulated boundary conditions and its load positions as described in Table 3.1. The magnitude of the horizontal and vertical components of the loads, after decomposition, varies between 0.5 kN and 4.33 kN. These loads are normalized between (100,0,0) and (255,0,0) as RGB colors to create a color image where the colored part represents the location and magnitude of the load as shown in Figs. 3.5c and 3.5d.

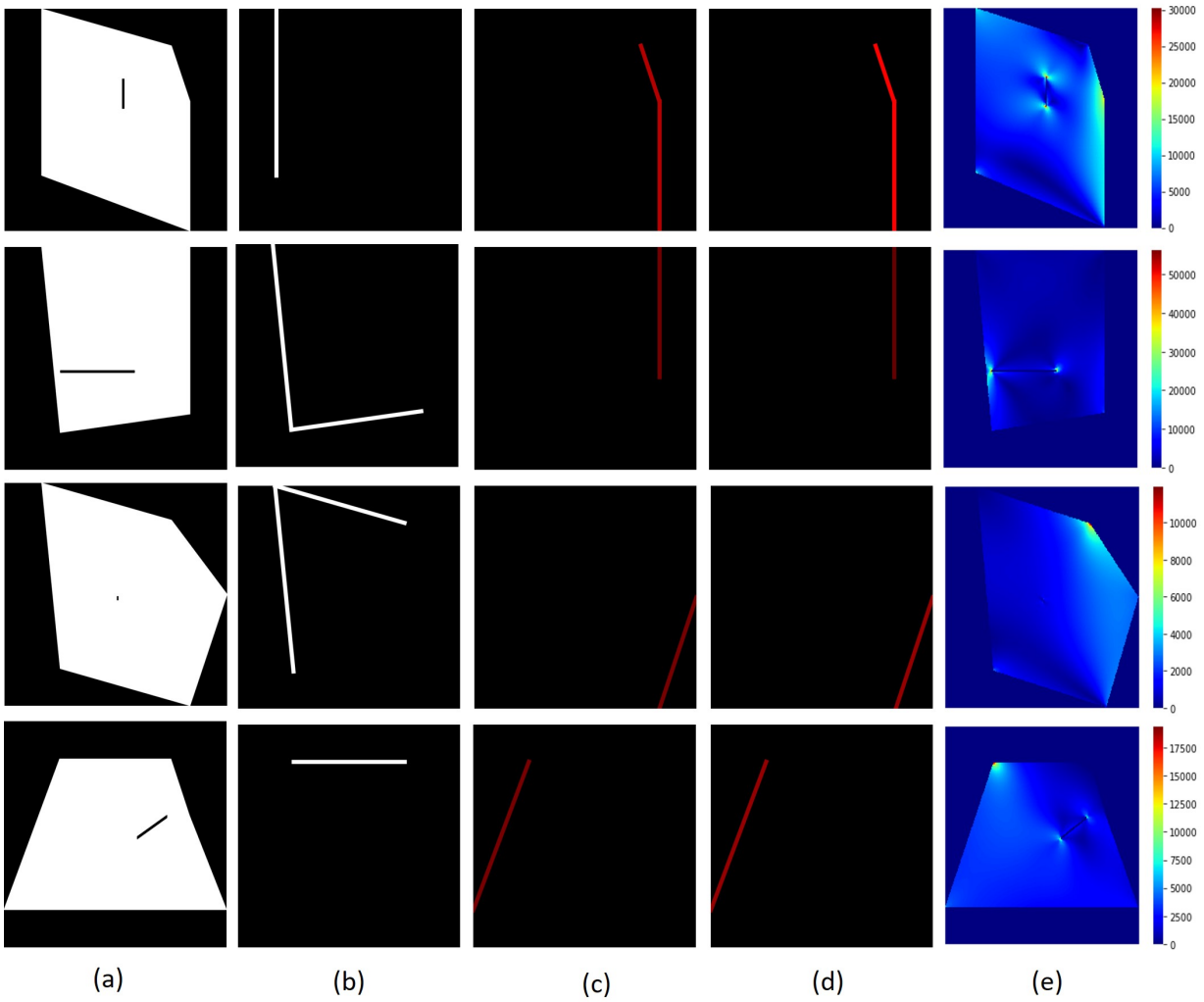


Figure 3.5 Input and output representation for stress distribution prediction: (a) damaged geometry, (b) boundary condition, (c) horizontal load, (d) vertical load, (e) output.

3.2.1.2 Output Data

FEA was performed using the Partial Differential Equation (PDE) solver in the MATLAB toolbox to obtain the stress distributions of each sample. We didn't use other FE approaches, such as a Carerra unified formulation [54] due to computational cost. Carrera Unified Formulation allows FE matrices and vectors to be derived in terms of fundamental nuclei. The MATLAB PDE toolbox mesh generator only generates unstructured triangulated meshes, which are not compatible with CNN. The minimum and maximum triangulated mesh sizes are 5 mm and 10 mm; respectively. Since each element should be represented by one pixel in an image, we develop a 600 × 600 grid surface equal to the dimensions of the largest possible geometry. Figs. 3.6a and 3.6b show the

unstructured mesh and 600 by 600 grid surface on top of the one random sample, respectively. The stress values are then interpolated between the triangular elements and grids to determine a stress distribution compatible with our CNN network. The stress values of all the elements outside of the material geometry are assigned to zero, as shown in Fig. 3.5e. The dimensions of the largest sample are 600 × 600 mm, and the smallest is 300 mm × 300 mm. Therefore, the dimension of each element is 1 mm × 1 mm, which means that each image has 360,000 pixels. All the cracks are initialized in the smallest dimension (300 × 300 mm) to keep all lengths of cracks inside the geometry. This high-resolution dataset led to achieving significant accuracy. The maximum and minimum von Mises stress values for elements among the entire dataset are 362,687 MPa and -138.35 MPa, respectively. We normalized all the output data between 0 and 1 to ensure faster convergence and encoded it to 600 × 600 matrices.

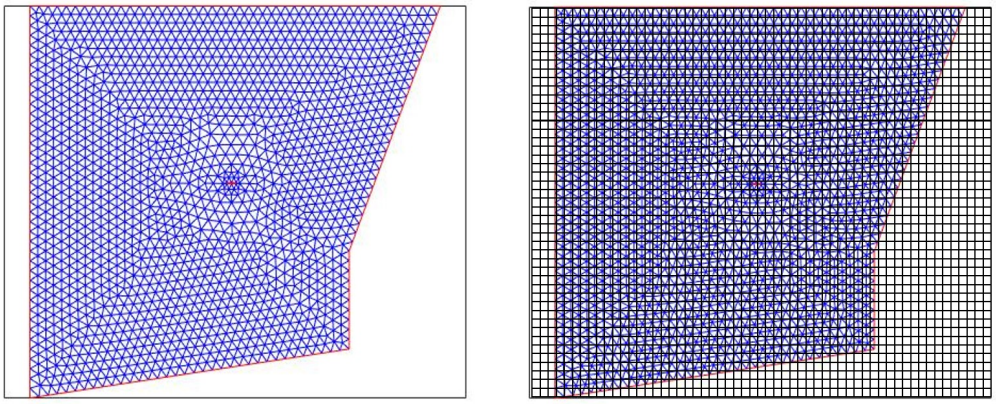


Figure 3.6 A sample of mesh generation: (a) unstructured triangular mesh, (b) structured gird surface.

3.2.2 CNN Architecture

The CNN can be built in different ways using a sequence of convolutional layers. The convolutional layers learn to encode the input in a set of simple signals and then reconstruct the input [45]. In Chapter 2, an ablation study was conducted to identify the most suitable architecture for the task. After careful consideration of the similarities between our dataset and the dataset used in Chapter 2, we decided to use an architecture that is almost identical to the one used in the ablation

study. The only difference is that our dataset includes cracks whereas the dataset in Chapter 2 does not. This was a conscious decision as the architecture used in Chapter 2 has been proven to work well for similar datasets. However, given the addition of cracks to our dataset, we decided to add self-attention layers to increase the weight of the crack area, which is of particular interest to us. Our CNN architecture consists of 4 types of layers: The first stage is downsampling layers which consist of seven convolutional layers (E1, E2, E3, E4, E5, E6, E7), and the second stage is 3 layers (RS1, RS2, and RS3) of Squeeze-Excitation and Residual blocks (SE-ResNet). The last stage is upsampling layers which consist of six deconvolutional layers (D1, D2, D3, D4, D5, D6) and 2 self-attention layers (SA1, SA2), as illustrated in Fig. 3.7. The layer sizes can be also seen in Table 3.3.

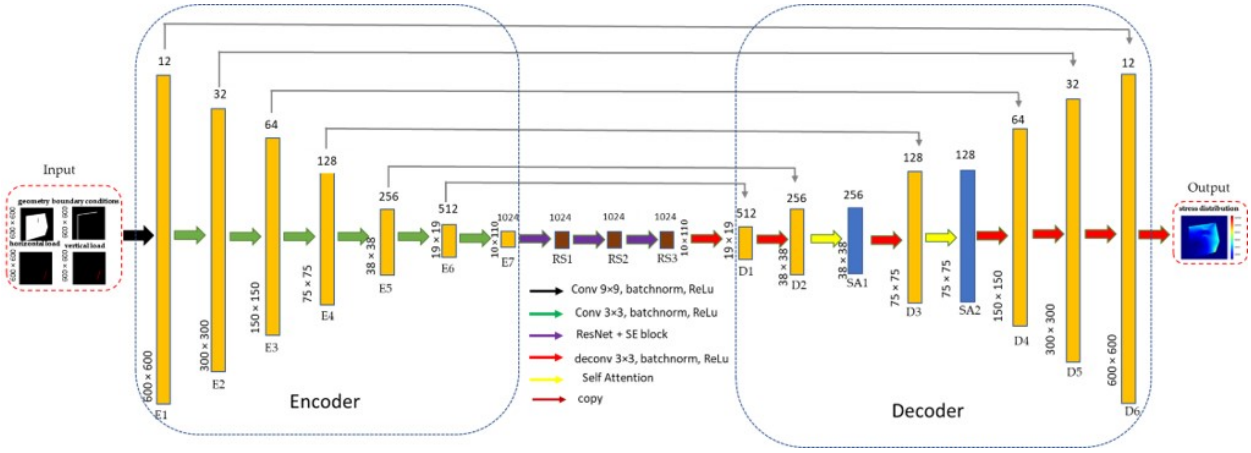


Figure 3.7 Model architecture.

Table 3.3 Size of network layers

Downsampling H×W×C	E1 600×600×12	E2 600×600×32	E3 600×600×64	E4 600×600×128	E5 600×600×256	E6 600×600×512	E7 600×600×1024	
SE-ResNet H×W×C	RS1 10×10×1024	RS2 10×10×1024	RS3 10×10×1024					
Upsampling H×W×C	D1 19×19×512	D2 38×38×256	SA1 38×38×256	D3 75×758×128	SA2 75×758×128	D4 150×150×64	D5 300×300×32	D6 600×600×12

3.2.2.1 Residual Block

We used residual blocks to address the vanishing gradient problem. In addition, SE blocks are computationally lightweight and result in only very small increases in model complexity. As illustrated in Fig. 3.8, the formulation of F(x)+x can be realized by feedforward neural networks

with shortcut connections. The shortcut connection simply performs identity mapping, and its output is added to the output of the stacked layers [46].

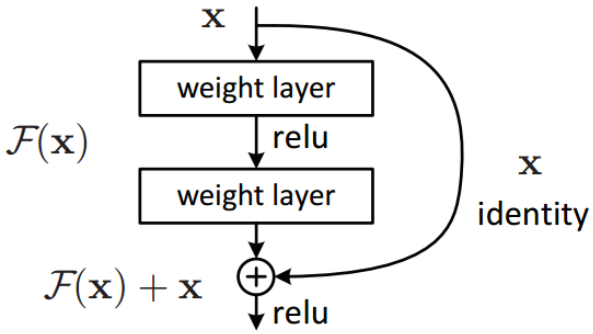


Figure 3.8 The building block of residual learning [46].

3.2.2.2 Squeeze-and-Excitation Blocks

As depicted in Fig. 3.9, Squeeze-and-Excitation blocks improve the representational capacity of the network, enabling dynamic channel-wise feature recalibration. A SE-block can be implemented with five steps. First, we feed the input x as a convolutional block and the current number of channels to the SE function, where F_{tr} in Fig. 3.9 is the convolutional operator for the transformation of X to U. Then, at the second phase, Each channel is squeezed into a single numeric value by using average pooling. Additionally, in the third phase, a fully connected layer is followed by a ReLU function, which applies a nonlinearity and reduces the output channel complexity. Then in the fourth phase, SE blocks can be used directly with residual networks. Fig. 3.10 depicts a SE-ResNet module in the SE block transformation. F_{tr} is regarded as the non-identity branch of a residual module. Before summation of the identity branch, both squeeze and excitation act. Using both SE and ResNet in the network outperforms using ResNet [47].

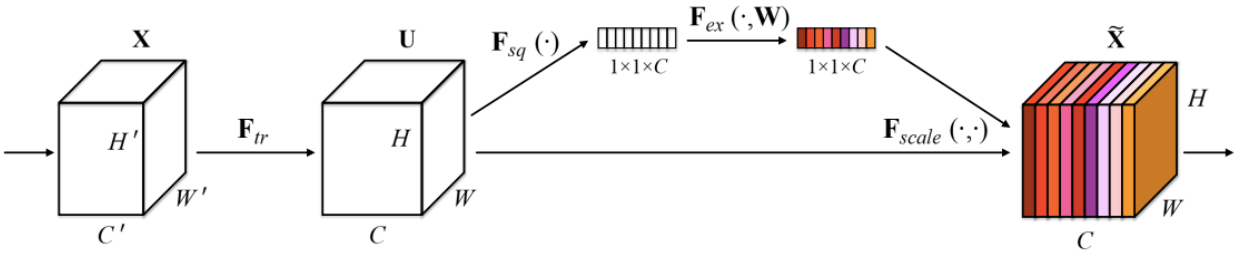


Figure 3.9 The building block of Squeeze-and-Excitation [47].

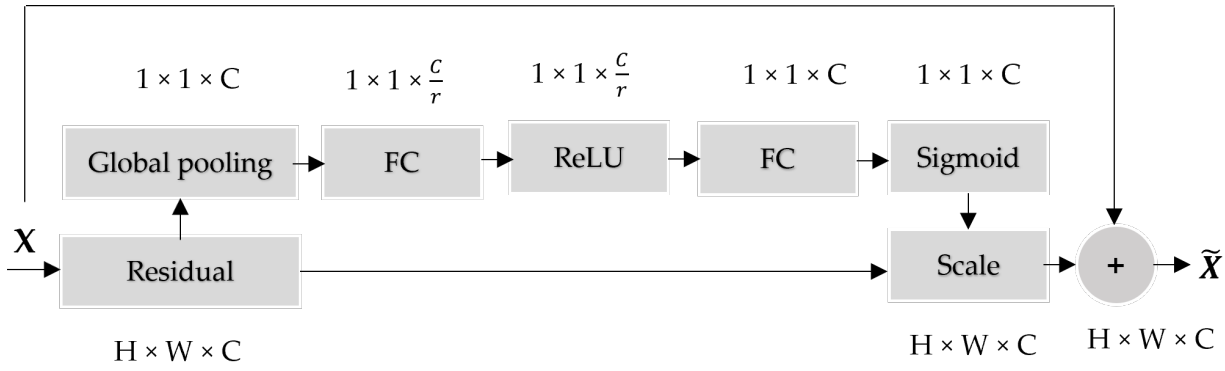


Figure 3.10 SE-ResNet module.

3.2.2.3 Self-attention blocks

In DL, the attention mechanism is inspired by human vision. Our brain transmits a signal via neurons after we receive visual information from the outside. Humans benefit from this process as it helps them focus on the right areas, and it reduces the weight of unrelated areas in their attention. As part of the feature extraction process of the input image, attention increases the weight of the area of interest and reduces the weight of unrelated regions. In the current paper, we use Self-Attention GAN (SAGAN) [55] to improve the prediction's results. Convolution processes information in a local neighborhood; therefore, using a single convolutional layer is computationally inefficient for modeling long-range dependency in images. SAGAN helps efficiently model relationships between widely separated spatial regions, even areas far apart; it can simply capture global dependencies. In the self-attentions mechanism, the convolutional image feature maps are broadened into three copies, corresponding to the key, value, and query concepts in the transformer [56]. key, value, and query are Key: $f(x) = W_{fx}$, Query: $g(x) = W_{gx}$, and Value: $h(x) = W_{hx}$. The image features from the previously hidden layer are first transformed into two feature spaces f(x) and g(x), to calculate the attention and then apply the dot-product attention to output the self-attention feature maps using equations 3.1 and 3.2. The entire process of the self-attention mechanism in SAGAN is depicted in Fig. 3.11.

$$a_{i,j} = Softmax(f(x_i)^T g(x_i)) \tag{3.1}$$

$$o_j = w_v(\sum_{i=1}^{n} a_{i,j} h(x_i)) \tag{3.2}$$

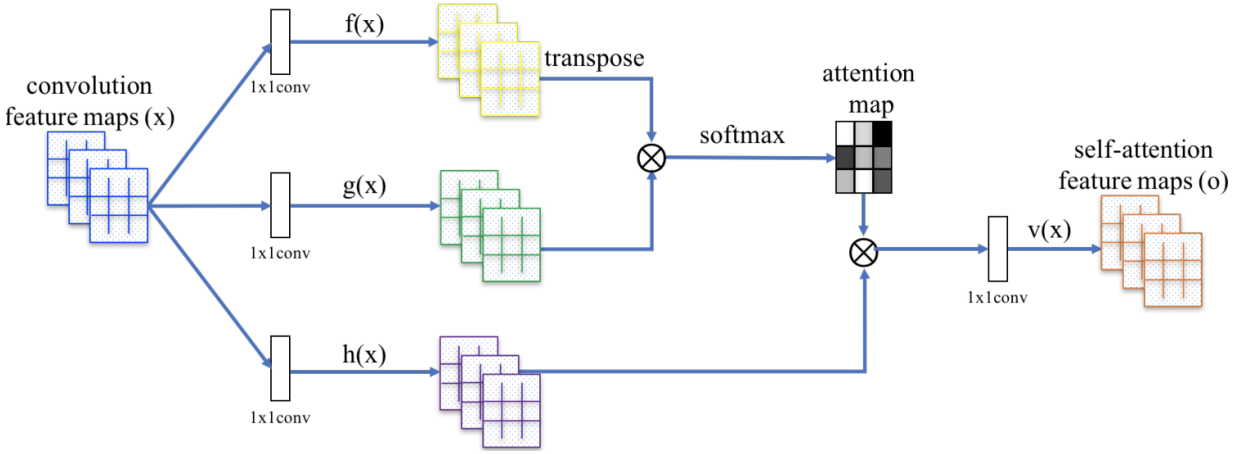


Figure 3.11 The building block of self-attention module for the SAGAN [55].

3.3 Loss function and performance metrics

3.3.1 Custom loss function

Learning biases can be established by proper choice of loss functions, constraints, and inference algorithms that can regulate the training step of the ML model to explicitly direct convergence towards solutions that adhere to the fundamental of physics [57]. The underlying physical laws can be satisfied by using and tuning such a penalty constraint. In this thesis, we consider the stress concentration factor equation to improve the prediction of stress concentration around the crack tip in steel gusset plates. It can be demonstrated from mathematical analysis and experimental results that stress distributions occur near changes in sections of a loaded structural component. It reaches greater magnitudes than the average stress in the section. It is called stress concentration when the peak stress increases near openings and other changes in the section. Equation 3.3 defines stress concentration as the peak stress relative to the nominal stress that would exist if the stress distribution remained uniform [58].

$$K_t = \frac{\sigma_{max}}{\sigma_{nom}} \tag{3.3}$$

where,K_t is the stress concentration factor, σ_{max} and σ_{nom} are peak stress around the crack tip and nominal stress in the remainder of the section, respectively. We create binary masks to apply the stress concentration factor equation to the loss function. Fig. 3.12 illustrates one of the possible mask scenarios, in Fig. 3.12a crack (rectangle with gray color) is surrounded by mask 1 (white rectangle) which all pixel values of the mask are one (white), and all other pixel values are zero (black). Fig. 3.12b shows mask 2 which all the zero pixel values from mask 1 are replaced with value one, and all one pixel's values are replaced with value zero. The relation between the two masks is mask 2 = 1 - mask 1. Mask 1 represents the area where the stress concentration factor should be applied to capture the peak stress, and mask 2 represents the area with nominal stress distribution. Based on the above description, our custom loss function will be defined as below:

$$Loss = \frac{\lambda_{PHY}}{n}\sum_{n=1}^{n}(S(i) - S^{\wedge}(i))^2 \cdot M_1 + \frac{1}{m}\sum_{m=1}^{m}(S(i) - S^{\wedge}(i))^2 \cdot M_2 \tag{3.4}$$

where λ_{PHY} is the stress concentration factor, n and m are the number of white pixels in mask 1 and mask 2, respectively. M_1 is mask 1 and M_2 is mask 2. S(i) is the stress value at a node 'i' computed by FEA as the ground truth and, s (i) is the corresponding predicted stress by the DL model, and '·' is the symbol of the Hadamard product.

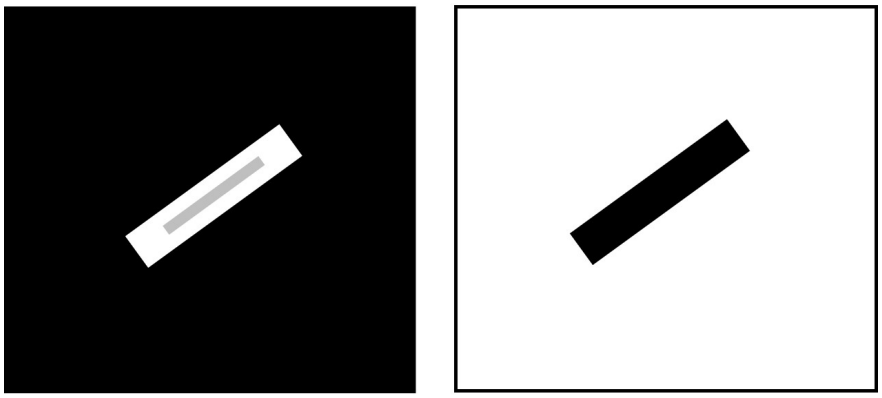


Figure 3.12 Illustration of binary masks. (a) mask 1, (b) mask 2.

3.3.2 Performance metrics

We used the custom loss function for the training loss as defined in equation 2. Therefore, its error is defined as CLE (Custom Loss Error). We also used MAE (Mean Absolute Error),

PMAE (Percentage Mean Absolute Error), PAE (Peak Absolute Error), and PPAE (Percentage Peak Absolute Error) to evaluate the overall quality of predicted stress distribution. These metrics are defined in Equations 3.5 3.6, 3.7, 3.8, respectively.

$$MSE = \frac{1}{n}|\sum_{i=1}^{n}(S(i) - S^{\wedge}(i))^2| \tag{3.5}$$

where $S(i)$ is the stress value at a node 'i' computed by FEA as the ground truth, $S^{\wedge}(i)$ is the corresponding predicted stress by the DL model, and n is the total number of elements at each sample which is 360,000 in our work. Symbol || denotes the absolute value. Our model's prediction and ground truth are displayed as 600×600 resolution images.

The percentage mean absolute error is defined as:

$$PMAE = \frac{MAE}{max[S(i)] - min[S(i)]} \times 100 \tag{3.6}$$

where max $[S(i)]$ is the maximum value in a set of ground truth stress values and min $[S(i)]$ is the minimum value.

PAE and PPAE measure the accuracy of maximum stress which are one of the main important critical load values in the predicted stress distribution. The importance of maximum stress matters in the design phase, since maximum stress should be less than yeild strength to avoid permanent deformation. PAE and PPAE are defined as:

$$PAE = [S(i)] - [S^{\wedge}(i)] \tag{3.7}$$

$$PPAE = \frac{PAE}{[S(i)]} \times 100 \tag{3.8}$$

3.4 Implementation details

All codes are written in PyTorch Lightning and run on two NVIDIA TITAN RTX 24G GPUs. AdamW optimizer [59] was used to speed up the convergence of our models using a learning rate of 1e−5. The batch size is set to 8, leading to the best accuracy compared to other batch sizes. The

value of the stress concentration factor, which is applied as a λ_{PHY} in the custom loss function is 15, leading to the best results compared to the other values.

3.5 Results and discussions

3.5.1 Main results

We train and evaluate our model on custom loss using the entire dataset for 200 epochs, the other metrics are plotted as independent metrics. The training data size is 49,152, and the separate testing data size is 12,288, which is randomly divided with a train/test ratio of 80% - 20%. Fig. 3.13 shows Custom Loss Function (CLE) and MAE losses as a function of epochs. Fig. 3.13a is in linear scales, and Fig. 3.13b is in logarithmic scales. Fig. 3.13a shows that the curves of both CLE and MAE rapidly declined after a few epochs. However, Fig. 3.13b gives a more precise representation of the model's behavior. From Fig. 3.13b, it can be seen that CLE is less than MAE, which is due to penalization of the loss with the stress concentration factor.

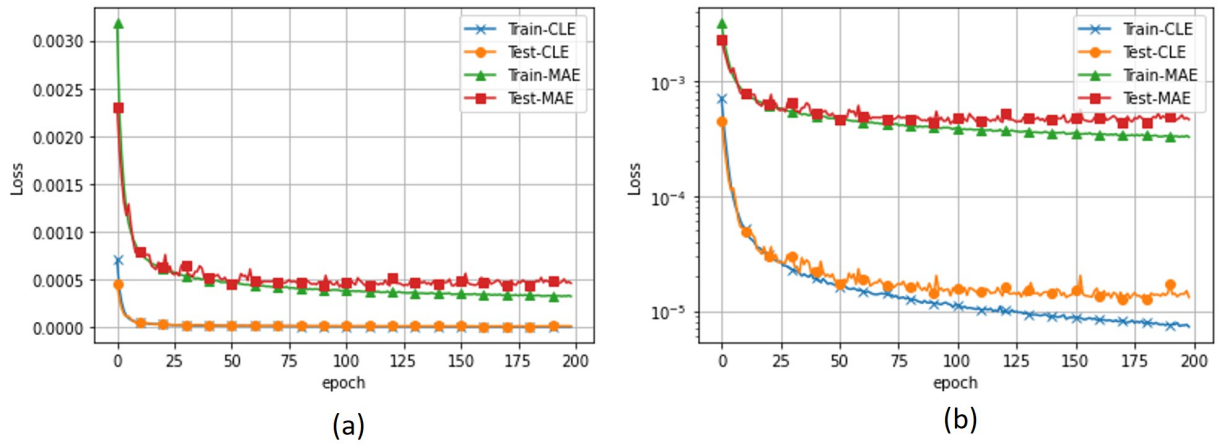


Figure 3.13 CLE and MAE curves on training and testing data with two scales: (a) linear scale, (b) logarithmic scale.

We saved the best checkpoint during training, epoch 181, and all error metrics are based on this checkpoint. The evaluation results of the network are shown in Table 3.4.

As can be seen, PMAE and PPAE for the testing dataset are just 0.22% and 1.50%, respectively. We consider these results satisfactory for stress distribution predictions of damaged structural components, specifically the PPAE, which is the most critical load value for stress distribution and

Table 3.4 Error metrics at epoch 181 (Units: MPa)

Metrics	CLE	MAE	PMAE(%)	PAE	PPAE(%)
Testing	4.5	22.01	0.22	10.8	1.5

stress concentration in engineering domain applications.

Fig. 3.14 illustrates the cumulative distribution of PMAE and PPAE in the test dataset. Fig. 3.14a shows that the probability of mean in PMAE is about 2%, which means that about 2% of predicted samples have a PMAE of less than 0.22; however, 80% of predicted samples have PMAE less than 3, and 50% of predicted samples have a PMAE of less than 1.21, which is the median. Fig. 3.14b shows that about 99% of predicted samples have a PPAE of less than 1.5, 80% of predicted samples have PMAE less than 0.35 and, 50% of the predicted samples have a PPAE of less than 0.17.

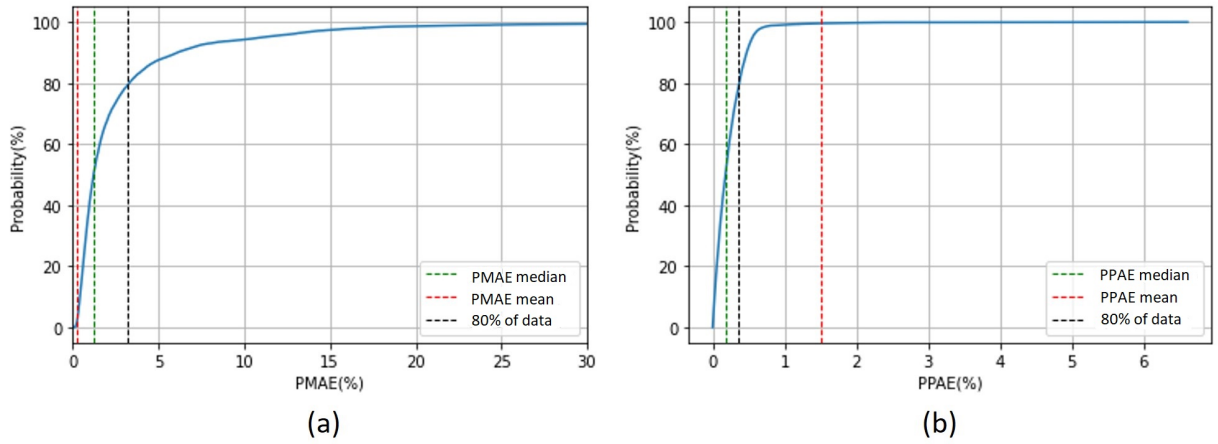


Figure 3.14 Cumulative distribution of PMAE and PPAE: (a) PMAE of samples less than mean, median and, 80% of data on the test dataset, (b) PPAE of samples less than mean, median and, 80% of data on the test dataset.

The prediction results of some randomly selected samples from the test dataset are visualized in Fig. 3.15. Each row represents a sample. Columns (a) to (d) represent geometry, boundary conditions, and load in horizontal and vertical directions, respectively. Columns (e) and (f) represent the ground truth and predicted stress distributions, respectively. Comparing columns (e) and (f) shows that predicted stress distributions are pretty similar to the ground truths.

Fig. 3.15 also demonstrates that the network can localize and quantify different sizes of damage, even tiny cracks. The second row of Fig. 3.15 is an example of rectangle damage with a size of $10 \times$

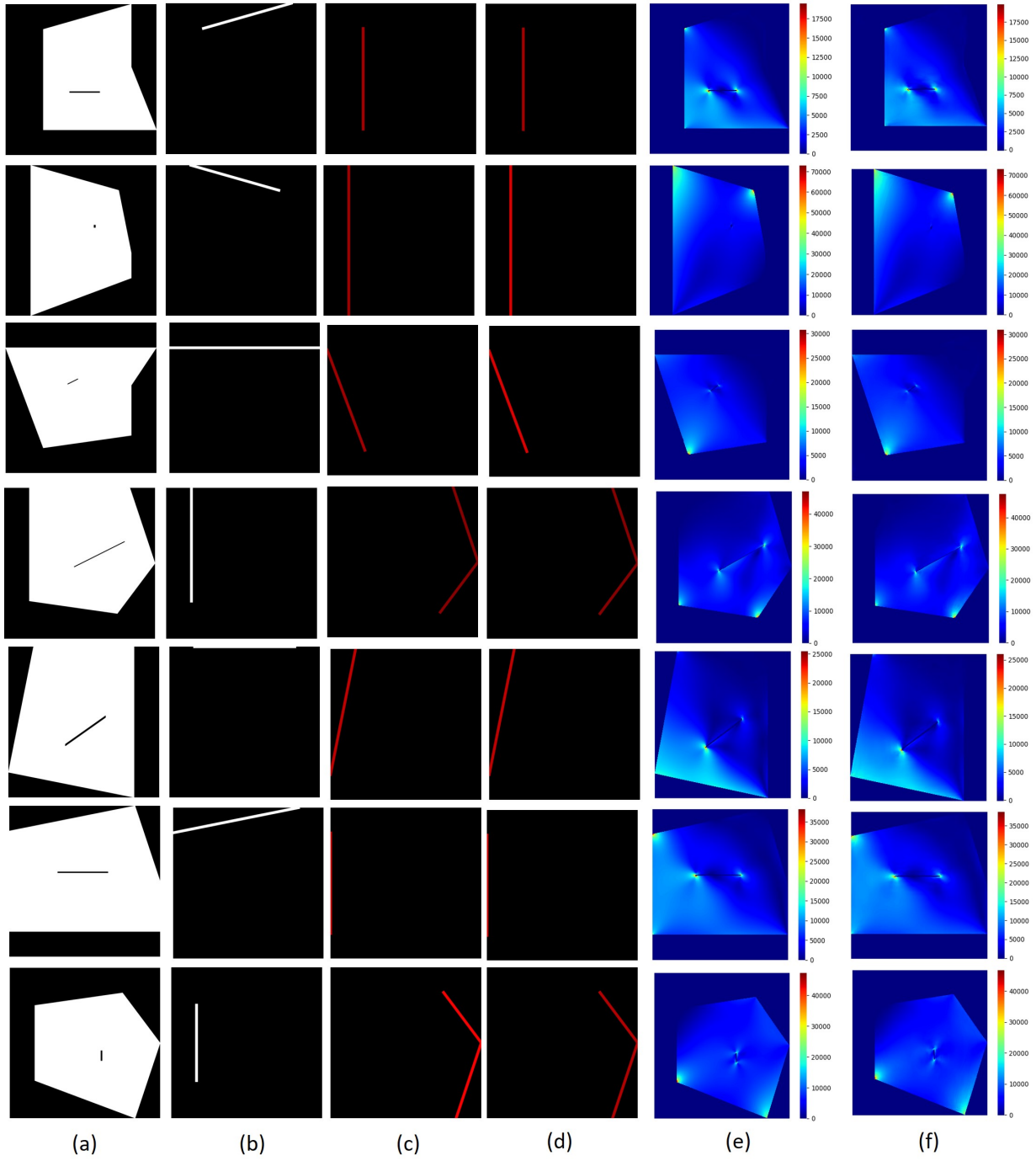


Figure 3.15 Predicted stress distribution and corresponding inputs with different loads and boundary conditions scenarios. Columns (a) to (d) represent (a) geometry, (b) boundary conditions, and load in a (c) horizontal and (d) vertical direction. Column (e) represent ground truth, and column (f) shows predicted stress distribution, respectively (Units =MPa).

1 mm, which shows proper damage localization and stress distribution prediction for the remainder of the section.

Fig. 3.16 shows two cases of Fig. 3.15 with larger scales to show details of ground truth and predicted stress distribution. Moreover, comparing the last two columns of Fig. 3.15 shows the efficacy of our novel custom loss function, which can accurately capture stress concentration around crack tips. This means the learned algorithm can capture the underlying knowledge of physics behind the stress concentration.

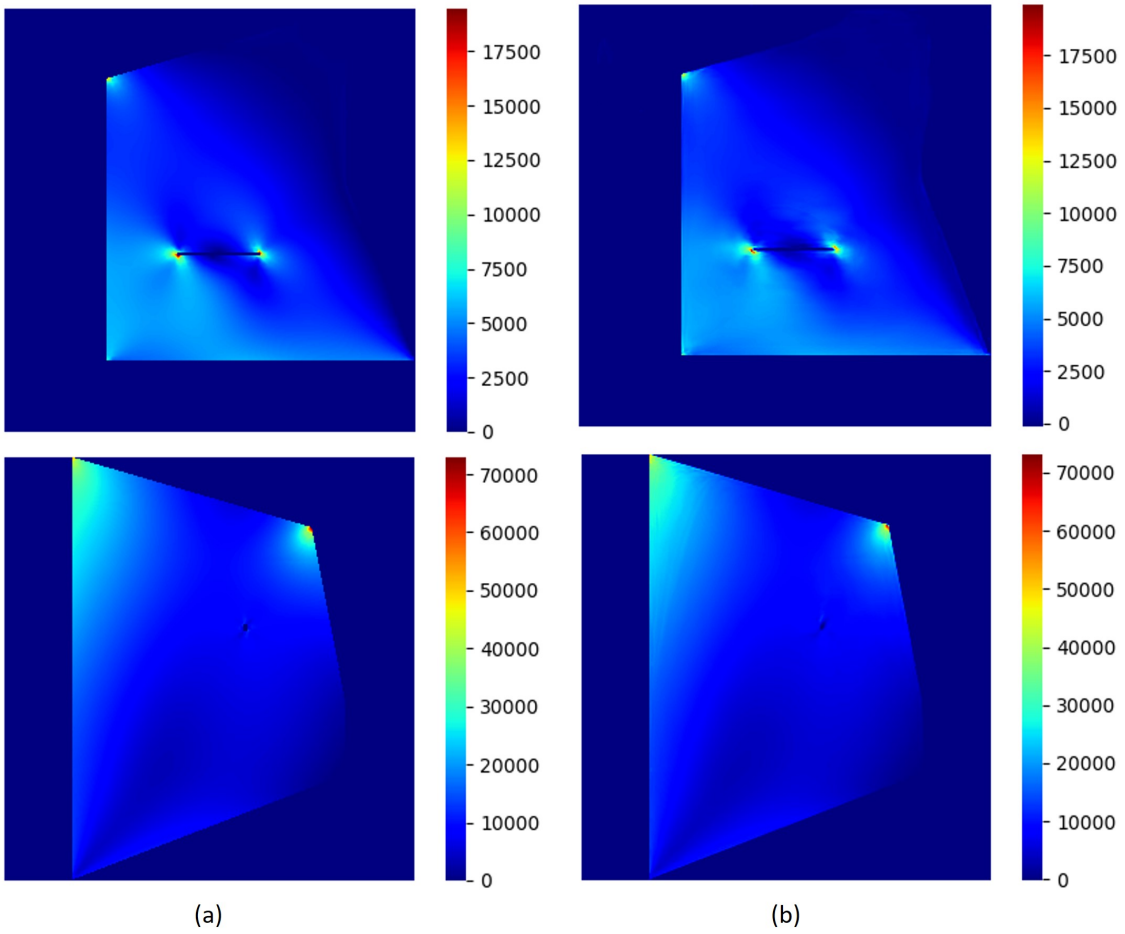


Figure 3.16 Larger representative from Fig. 15. (a) ground truth (b) predicted (Units = MPa).

Some inaccurate predictions are also shown in Fig. 3.17. As can be seen, these predictions can still capture damage locations and stress concentration around crack tips; however, mean and peak stress distribution in some parts of the ground truth and predictions slightly vary. Fig. 3.18 shows two cases of Fig. 3.17 with larger scales to show details of ground truth and predicted stress distribution.

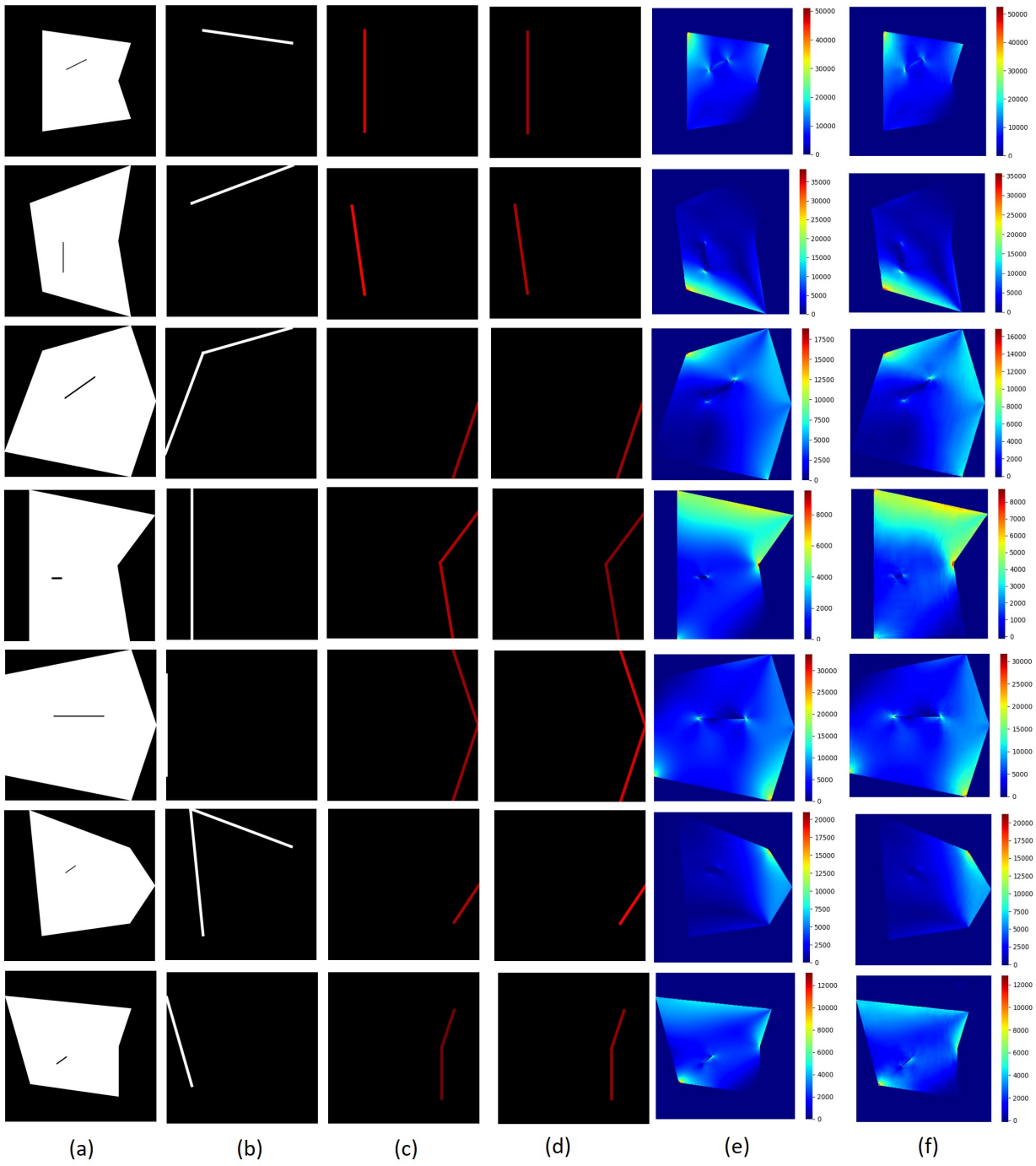


Figure 3.17 Inaccurate predicted stress distribution and corresponding inputs with different loads and boundary conditions scenarios. Columns (a) to (d) represent geometry, boundary conditions, load in the horizontal and vertical direction, respectively. Columns (e) and (f) represent ground truth and predicted stress distribution, respectively (Units = MPa).

3.5.2 Study on the effect of the custom loss function

We have also trained our model using torch.nn.MSELoss function compares its results with our custom loss model to evaluate how efficient our proposed custom loss function is. We investigated

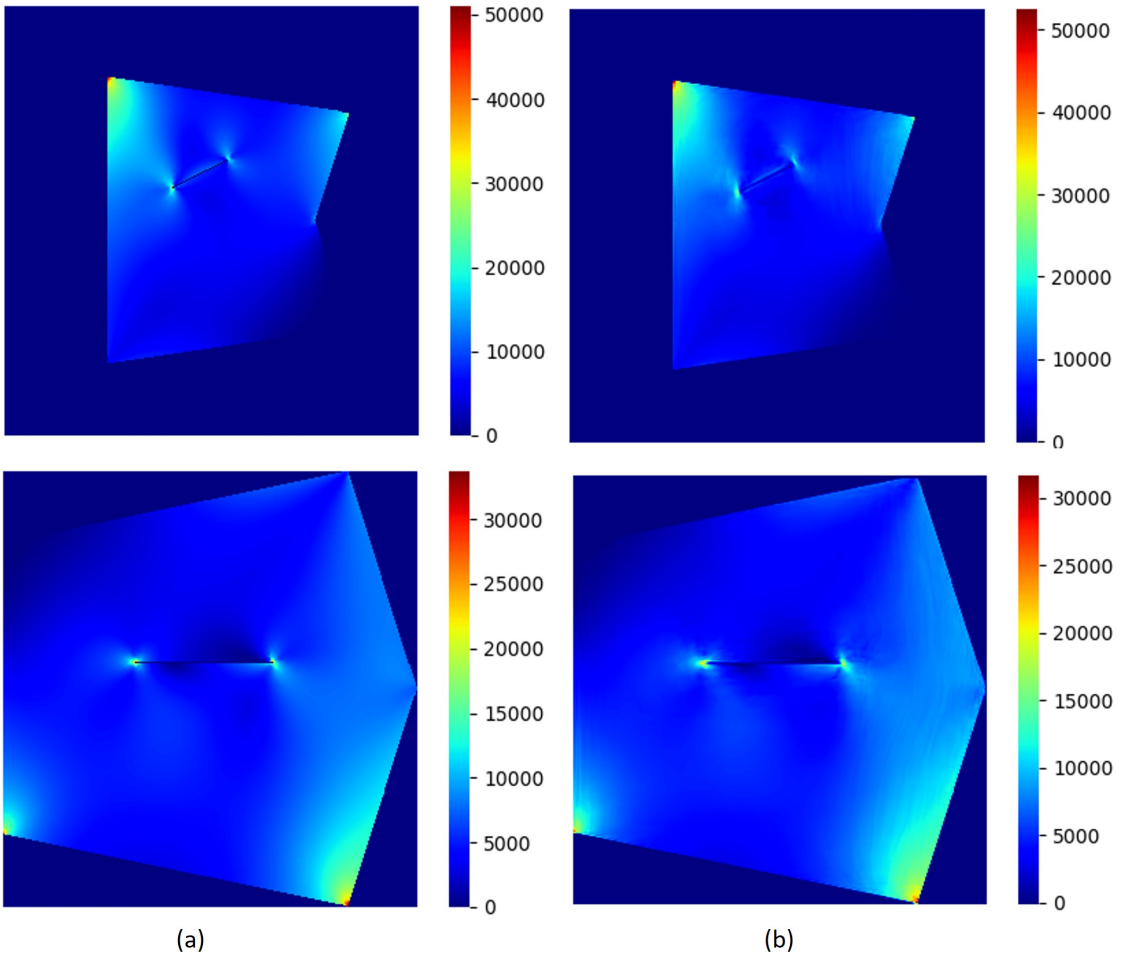


Figure 3.18 Larger representative from Fig. 17. (a) ground truth (b) predicted (Units = MPa).

the performance of the custom loss model with different stress concentration factors and compared it with the MSE model. Table 3.5 demonstrates that the performance of the custom loss model with a stress concentration factor equal to 1 is almost the same as the MSE model, which is expected based on equation 3.4.

Table 3.5 Error metrics with different stress concentration factors (Units: MPa)

Model	Stress concentration factor	PMAE(%)	PPAE(%)
Custom loss	1	0.05	2.52
Custom loss	8	0.12	2
Custom loss	15	0.22	1.5
Custom loss	20	0.27	2.3
Custom loss	30	0.48	3.7
MSE loss	None	0.053	2.5

However, with applying higher and lower values than 15 the PPAE increased in the custom loss model. It seems lower stress concentration factors do not trigger the pixel values within the damaged area and higher stress concentration factors are over triggering the pixel values within the damaged area. Therefore, we trained our custom loss model with stress factor 15 to obtain the best results. The error metric in the MSE loss model and the custom loss model with stress factor 15 are presented in the Table 3.6.

Table 3.6 Error metrics in MSE and custom loss model (Units: MPa)

Metrics	CLE	MAE	PMAE(%)	PAE	PPAE(%)
Custom loss	4.5	22.01	0.22	10.8	1.5
MSE		5.07	0.053	109.03	2.5

As can be seen, the custom loss model's performance is slightly better than the MSE model in terms of PPAE, which is expected since we penalize the damaged areas 15 times more than the remainder of the section. However, the MSE model performs better in terms of the MAE metric than the custom loss model. This means the custom loss model is better in damage localization and capturing stress concentration around the crack tips, and the MSE model is better in general stress prediction. Although PPAE in the custom loss model is 1% less than the MSE model, the custom loss has a significant advantage in damage localization and capturing stress concentration around crack tips.

Fig. 3.19 compares predicted stress distribution between MSE and custom loss model and the masks used for the custom loss function. In Fig. 3.19, Columns (a) represent the mask used for penalizing the damaged area, and column (b) represents the mask used for the remainder of the section, which has no penalization. Columns (c), (d), and (e) represent the ground truth, MSE model prediction, and, custom loss model prediction. As it can be seen in the first row of Fig. 3.19, the MSE model cannot localize the crack; however, the custom loss model has completely localized the tiny crack and its stress concentration around the crack tip. Fig. 3.19 also shows that the MSE model can localize larger cracks but still is not as good as the custom loss model in capturing stress concentration around crack tips. Fig. 3.20 shows two cases of Fig. 3.19 with larger scales to show

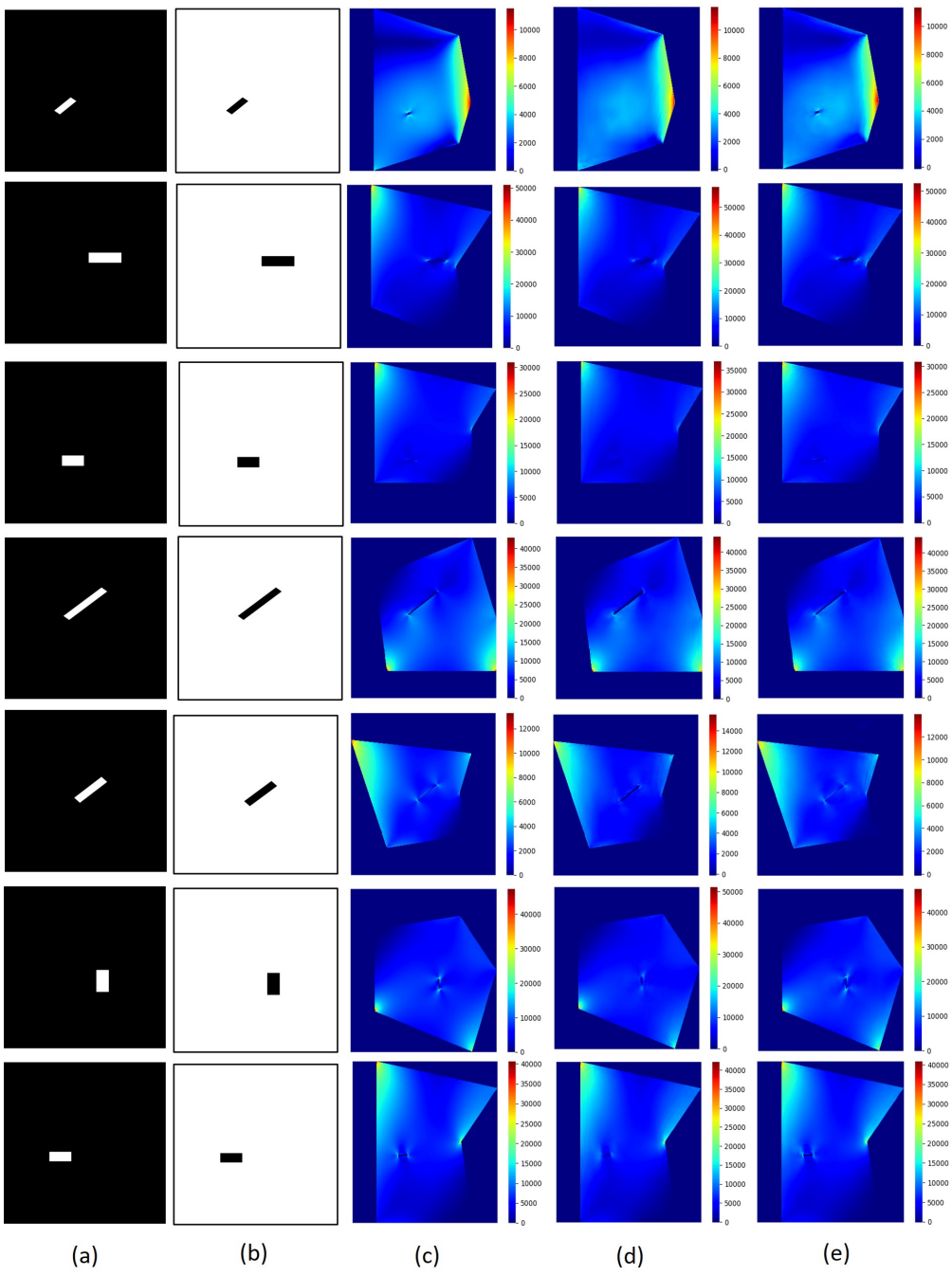


Figure 3.19 Comparison of predicted stress distribution in MSE and custom loss model and their corresponding masks. Columns (a) and (b) represent the mask for the damaged area and the remainder of the section, respectively. Columns (c) to (e) represent ground truth, and predicted stress distribution in the MSE and custom loss models, respectively (Units = MPa).

details of ground truth and predicted stress distribution with the MSE loss model and custom loss model.

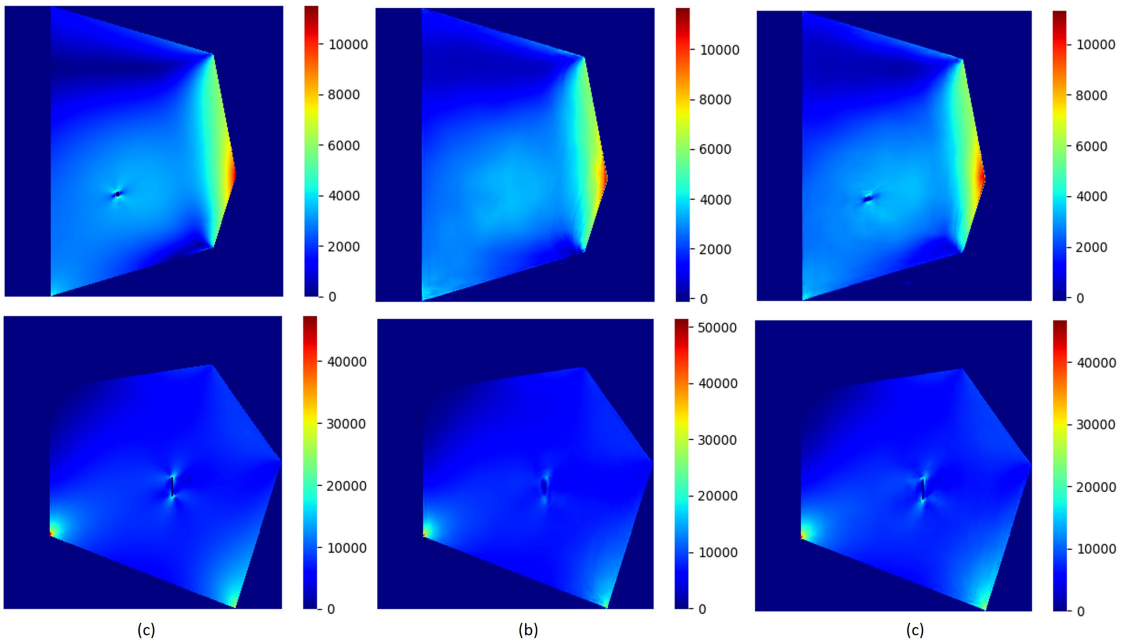


Figure 3.20 Larger representative from Fig. 19. (a) ground truth (b) MSE loss model (c) custom loss model (Units = MPa).

CHAPTER 4

NEURO-DYNASTRESS: PREDICTING DYNAMIC STRESS DISTRIBUTIONS IN STRUCTURAL COMPONENTS

4.1 Need for fast dynamic analysis

Structural components are typically exposed to dynamic loadings, such as earthquakes, wind, and explosions. Structural engineers should be able to conduct real-time Finite Element Analysis (FEA) aftermath or during extreme disaster events requiring immediate corrections to avoid fatal failures. For instance, if after a disruptive event a crack occurs in the bridge, fast FEA will help engineers to understand what kind of immediate action should be taken before the failure of the bridge. As a result, it is crucial to predict dynamic stress distributions during highly disruptive events in real time. The main idea here is to train a model that can later be used when real-time estimations are needed, such as in the aftermath of extreme disruptive events. For example, focusing on critical structural components, there is a need for immediate assessment following a disaster or during extremely disruptive events to guide corrective actions. Engineers could rely on the proposed computationally efficient algorithms to determine stress distributions over damaged gusset plates and apply the proper rehabilitation actions. They need to be able to analyze gusset plates quickly and accurately, which is what our model can provide.

Some studies have been conducted to develop methods of predicting structural response using ML models. Dong et al. [32] proposed a support vector machine approach to predict nonlinear structural responses. Wu et al. [33] Utilized deep convolutional neural networks to estimate the structural dynamic responses. Long short-term memory (LSTM) [34] was used by Zhang et al. [35] to predict nonlinear structural response under earthquake loading. The few models that studied stress predictions suffer from the problem of low-resolution predictions, making them unsuitable for decision-making after a catastrophic failure. To the best of our knowledge, this is the first work to predict dynamic stress distribution in the specific domain of steel plates with high accuracy and

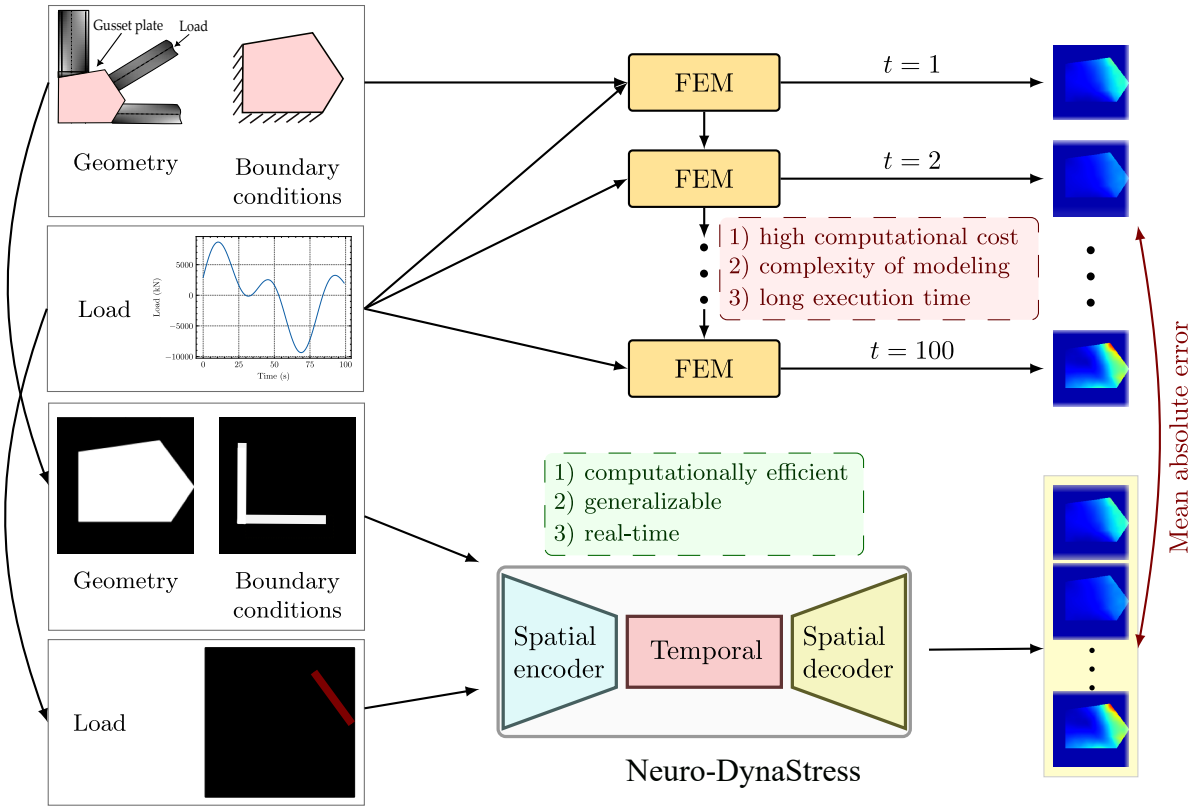


Figure 4.1 **Overview:** Unlike FEM, our proposed Neuro-DynaStress is computationally efficient and facilitates real-time analysis. The existing workflow for FEM applications includes: (i) modeling the geometry and its components, (ii) specifying material properties, boundary conditions, meshing, and loading, (iii) dynamic analysis, which may be time-consuming based on the complexity of the model. Our Neuro-DynaStress takes geometry, boundary condition, and load as input and predicts the dynamic stress distribution at all time steps in one shot.

low latency. The algorithm takes the geometry, boundary conditions, and time histories as input and renders the dynamic von Mises stress distribution as an output. We modeled the steel plates as gusset plates with dynamic loading applied at different edges, different boundary conditions, and varying complex geometries.

4.2 Methods

4.2.1 Data Generation

The data generation process is almost identical to its representation in chapter 2, and we have included it in this chapter as well since each chapter discusses a different topic. Two-dimensional steel plate structures with five edges, E1 to E5 denoting edges 1 to 5, as shown in Fig. 4.2, are

considered homogeneous and isotropic linear elastic materials. Various geometries are generated by changing the position of each node in horizontal and vertical directions, as shown in Fig. 4.2, which led to 1024 unique pentagons. The material properties remain unchanged, isotropic for all samples. The 2D steel plates approach the geometry of gusset plates. 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. The boundary conditions and time-history load cases are considered to simulate similar conditions in common gusset plate structures under external loading. We showed the most common gusset plates in practice in chapter 2.

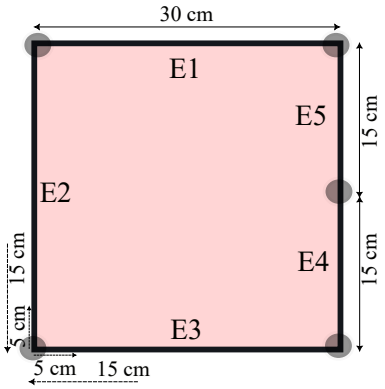


Figure 4.2 Basic schematic topology for initializing the steel plate geometries.

A total of 57,344 unique samples were created by combining 14 random time-history load cases and the four most common boundary conditions in gusset plates. Boundary conditions are shown in Fig. 4.3, 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. Each time history consists 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. 4.4. Considering 100 time steps, each interval is 0.01 seconds, making the total time equal to 1 second. All the details for the input variables used to initialize the population are shown in Table 4.1.

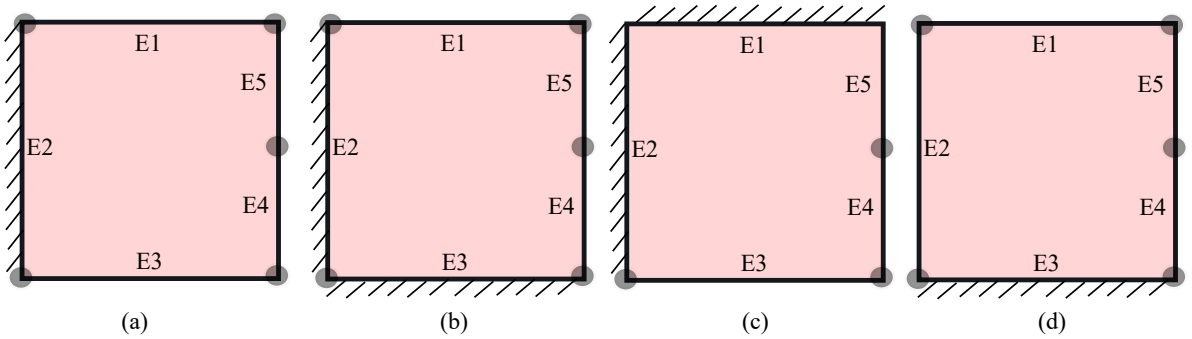
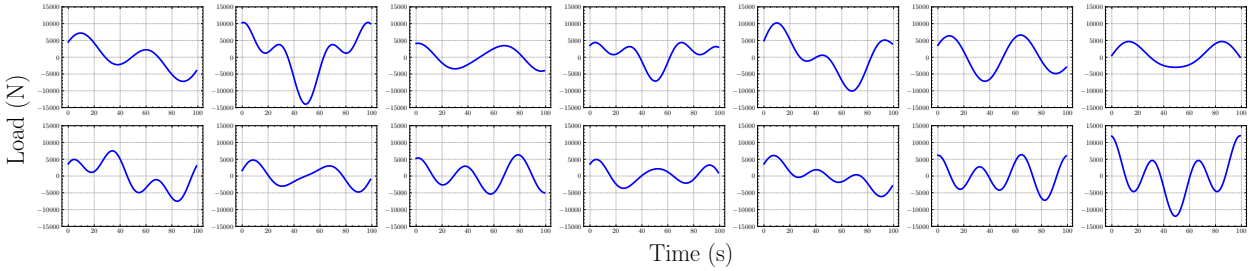
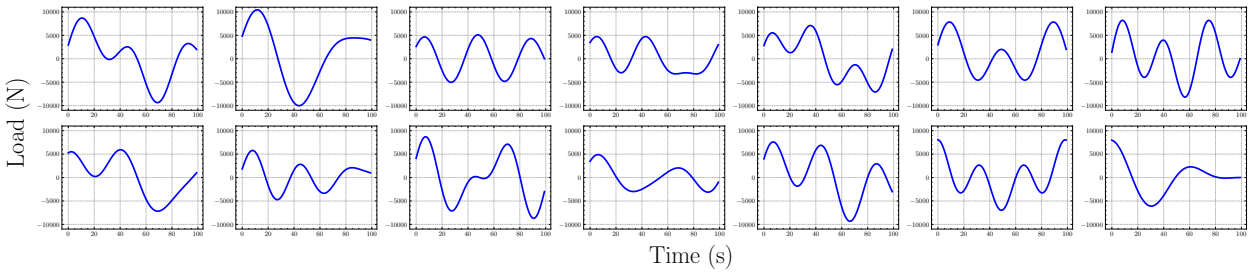


Figure 4.3 Different types of boundary conditions for initializing population.



(a)



(b)

Figure 4.4 Time histories (a) Horizontal direction (b) Vertical direction.

Table 4.1 Input variable

Geometry	Boundary conditions	Load position	Frequencies (HZ)	Load (kN)	Time steps	Total time (s)
pentagon	E2	E4E5	1,1.5,2,2.5,3	2,4,6,8,10	100	1
pentagon	E2E3	E5	1,1.5,2,2.5,3	2,4,6,8,10	100	1
pentagon	E1E2	E4	1,1.5,2,2.5,3	2,4,6,8,10	100	1
pentagon	E3	E2E5	1,1.5,2,2.5,3	2,4,6,8,10	100	1

4.2.2 Input Data

The geometry is encoded as a 200×200 matrix and, incidentally, a binary image. 0 (black) and 1 (white) denote outside and inside of the geometry, as shown in Fig. 4.5a. The boundary condition

is also represented by another 200×200 pixel binary image, where the constrained edges are defined by 1 (white) as shown in Fig. 4.5b. Moreover, each time step of time histories for horizontal and vertical components is encoded in the load position of the corresponding frame. Load positions in each time step have values between 0 and 1, corresponding to each time step of time histories, and all remaining elements are zero. All the load frames of each sample in horizontal and vertical directions are saved as tensors of dimension $100 \times 200 \times 200$. Figs. 4.5c and 4.5d show loads in the horizontal and vertical directions. The colored load positions in Figs. 4.5c and 4.5d are used only for visualization. Each row of Fig. 4.5 represents one of the simulated samples. Details of boundary conditions and their load positions are described in Table 4.1.

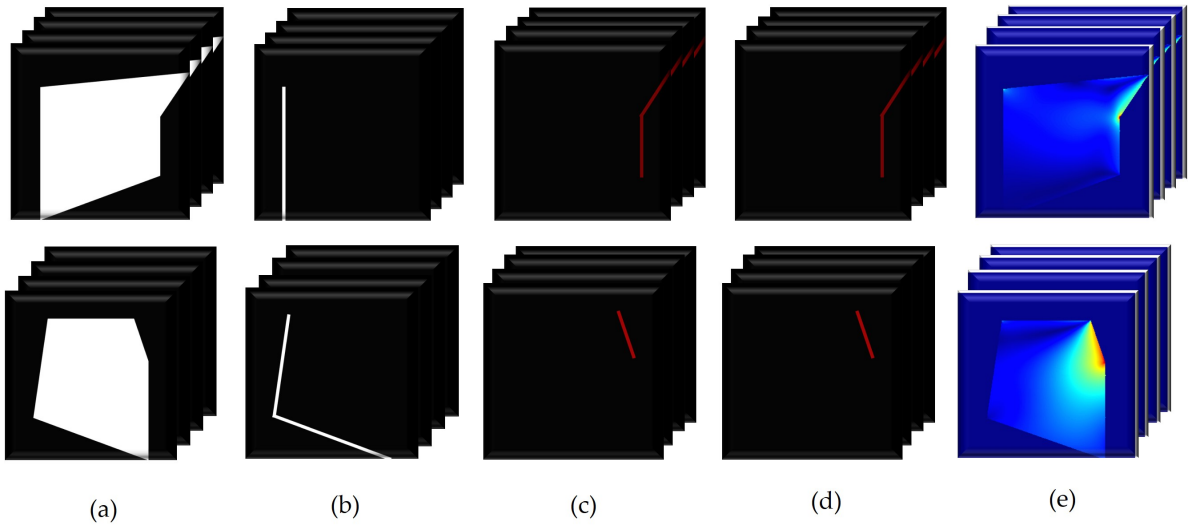


Figure 4.5 Input and output representation for stress distribution prediction: (a) geometry, (b) boundary condition, (c) horizontal load, (d) vertical load, (e) output.

4.2.3 Output Data

FEA was performed using the Partial Differential Equation (PDE) solver in the MATLAB toolbox to obtain the stress distributions of each sample. We used transient-planestress function of MATLAB PDE solver to generate dynamic stress contours as the ground truth of our model. We defined the geometry, boundary condition, material properties, and time histories as input, and the PDE solver returns the sequence of stress distributions corresponding to the inputs. The MATLAB PDE toolbox mesh generator only generates unstructured triangulated meshes incompatible with CNN. The minimum and maximum triangulated mesh sizes are 5 and 10mm, respectively. Since

each element should be represented by one pixel in an image, we develop a 200×200 grid surface equal to the dimensions of the largest possible geometry. Figs. 4.6a and 4.6b show the unstructured mesh and the 200×200 grid surface on top of a random sample. The stress values are then interpolated between the triangular elements and grids to determine a stress distribution compatible with our CNN network. The stress values of all the elements outside the material geometry are assigned to zero, as shown in Fig. 4.5e.

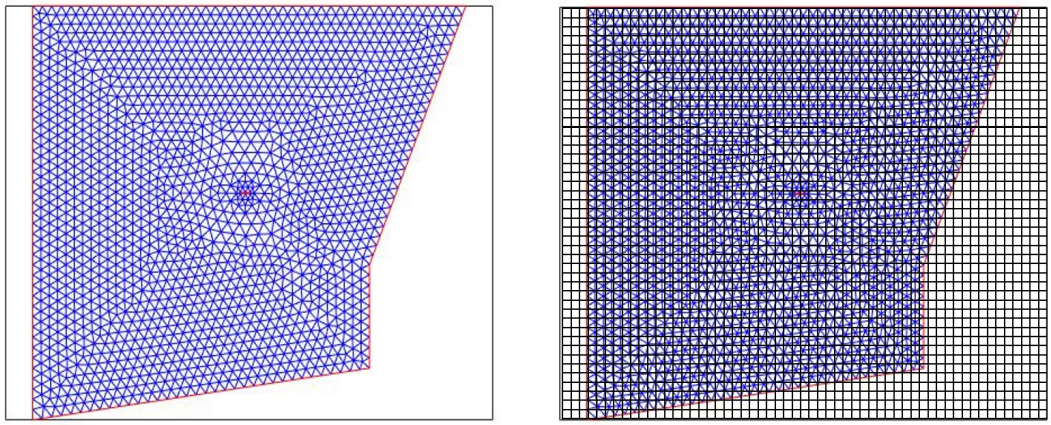


Figure 4.6 A sample of mesh generation: (a) unstructured triangular mesh, (b) structured gird surface.

The dimension of the largest sample is 600×600 mm, and the smallest is 300×300 mm. Using a mesh grid of 200×200 on top of samples made each element 3×3 mm, which means that each frame of output has 40,000 pixels. This high-resolution dataset led to achieving significant accuracy. The maximum and minimum von Mises stress values for elements among the entire dataset are 279,370 and -980 MPa, respectively. We normalized all the output data between 0 and 1 to ensure faster convergence and encoded it to 200×200 for each frame.

4.2.4 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:

$$KQ = F \tag{4.1}$$

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:

$$K_e = A_e B^T D B \tag{4.2}$$

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 4.1, we can calculate the nodal displacements q then we can calculate the stress tensors of each element as follows:

$$\sigma = DBq \tag{4.3}$$

where σ specifies the tensor of an element. The 2-D von Mises Stress criterion is then used to calculate each element's von Mises Stress:

$$\sigma_{v_m} = \sqrt{\sigma_x^2 + \sigma_y^2 - \sigma_x \sigma_y + 3\tau_{x_y}^2} \tag{4.4}$$

where σ_{v_m} denotes von Mises Stress, σ_x, σ_y are the normal stress components and τ_{x_y} is the shear stress component.

4.3 Proposed Methodology

We use convolutional layers to encode the spatial information from the input. Our hypothesis is that these layers will combine the information in geometry, boundary conditions, and load. A key characteristic of dynamic structural systems is the temporal dependence of their states. LSTM is a suitable architecture for modeling temporal information in sequence and hence is a good choice to model structural dynamic systems in our experiments. For high-quality 2D reconstructions, we use transposed convolutional layers in our decoder. For further improving training and performance, we use modules from the recently proposed feature-aligned pyramid networks (FaPN) [60]. FaPN allows the decoder to access information from the encoder directly. Overall, our network architecture

consists of four modules: encoder consisting of convolutional layers, temporal module made using LSTM modules, decoder consisting of transposed convolutional layers, and alignment modules acting as connections between encoder and decoder. The number of layers in each module and the number of layers in LSTM modules were chosen based on their performance. The architecture is illustrated schematically in Fig. 4.7. The size of layers and hyper-parameters used in the network are summarized in Table 4.2.

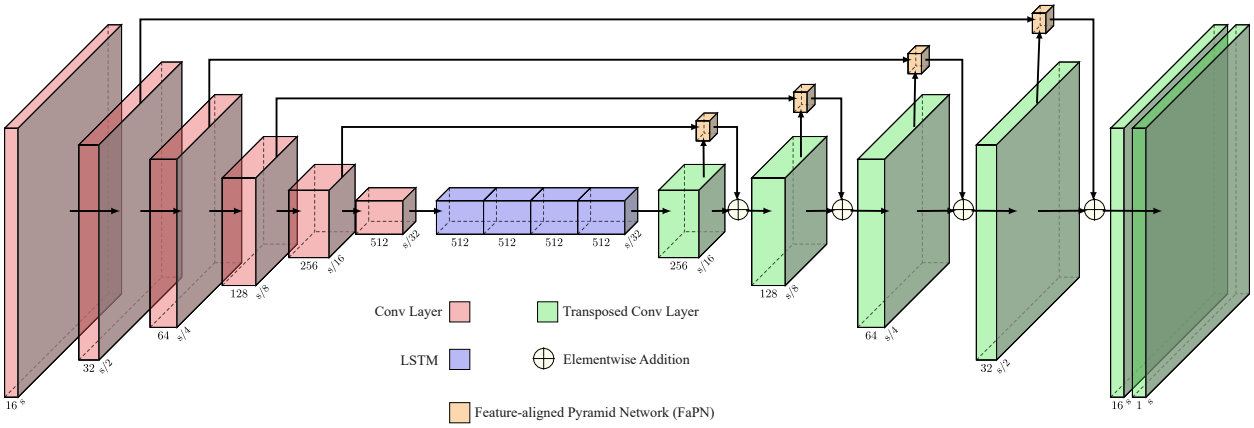


Figure 4.7 Architecture for the proposed Neuro-DynaStress. The convolutional encoder maps the raw input data to a latent space. LSTM layers processes the information across different time frames. The final output is obtained from the resulting latent representation using transposed convolutional layers.

Table 4.2 Network layers and hyper-parameters

Type of layers	Number of layers	First layer (H×W×C)	Last layer (H×W×C)
Conv	6	200×200×16	7×7×512
LSTM	4	1×1×512	1×1×512
ConvT	5	13×13×256	200×200×16
FaPN	4	13×13×256	100×100×32
Batch size	**Learning rate**	**Weight decay**	**Loss function**
8	10^{-4}	10^{-5}	MAE

4.4 Loss Function and Performance Metrics

We use Mean Absolute Error (MAE), defined in Eq. 4.5 as the primary training loss and metric. To ensure that we do not overfit to a single metric.

$$\text{MAE} = \frac{1}{NT} \sum_{N,T}^{n,t} \left| S(n,t) - \hat{S}(n,t) \right| \tag{4.5}$$

We also use Mean Relative Percentage Error (MRPE) defined in Eq. 4.6 to evaluate the overall quality of predicted stress distribution.

$$\text{MRPE} = \frac{\text{MAE}}{\max(|S(n,t)|, |\hat{S}(n,t)|)} \times 100 \tag{4.6}$$

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.

4.5 Implementation and Computational Performance

We implemented our model using PyTorch [61] and PyTorch Lightning. AdamW optimizer [59] was used as the optimizer with a learning rate of 10^{-4}. We found that a batch size of 8 gave the best results. The computational performance of the model was evaluated on an AMD EPYC 7313 16-core processor and two NVIDIA A6000 48G GPUs. The time required during the training phase for a single sample with 100 frames and a batch size of 8 was 10 seconds. In the training phase, one forward and backward pass was considered. The inference time for one sample was less than 5 ms which can be considered a real-time requirement. The most powerful FE solvers take between 10 minutes to an hour to solve the same. Therefore, Neuro-DynaStress is about 72×10^4 times faster than conventional FE solvers. We consider the minimum time for all processes of modeling geometry, meshing, and analysis of one sample in FE solver to be about 10 minutes. MATLAB PDE solver does not use GPU acceleration. This demonstrates that our proposed approach can achieve the real-time requirement during the validating phase.

4.6 Results and Discussions

4.6.1 Quantitative Evaluation

Our model is trained on the training dataset for 45 epochs and evaluated on the validation dataset using separate metrics. The training dataset consisted of 48,755, while the validating dataset contained 8,589 samples, together forming the 80%-20% split of the whole dataset. The

model predicts five frames of output from a sequence of five previous inputs until all 100 frames are predicted. The best validation performance was obtained when we sequenced five frames during validation. The best checkpoint during validation, at epoch 40, is the basis for all error metrics. MRPE for the validating dataset is just 2.3%.

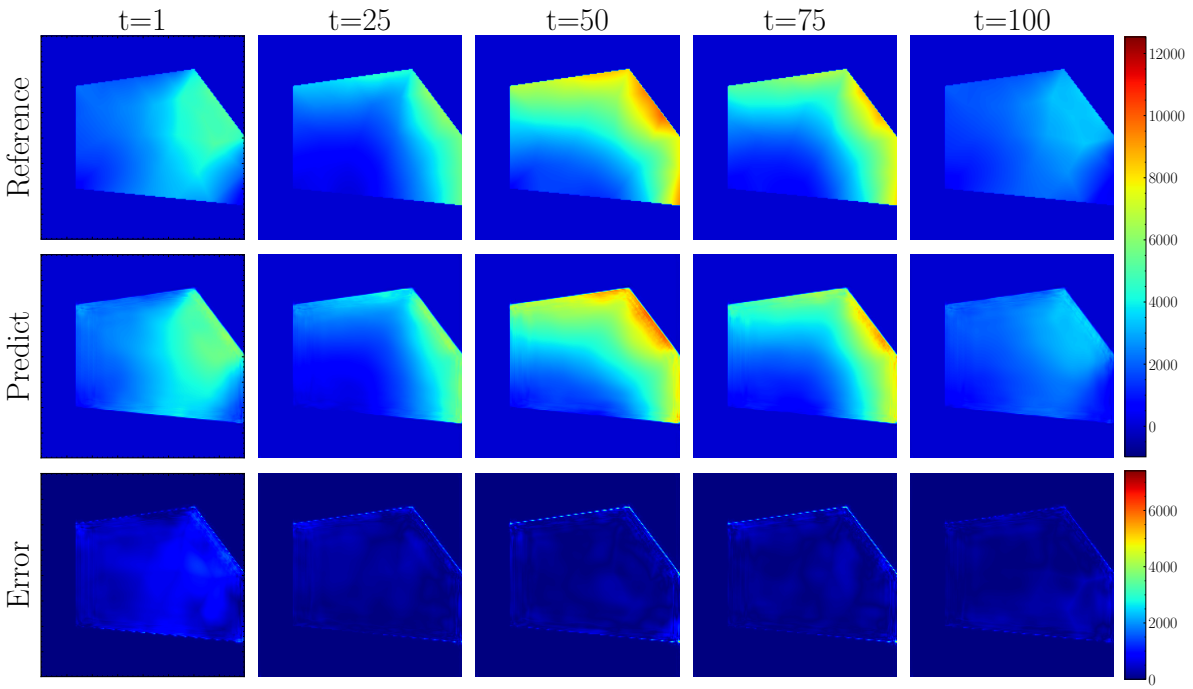
4.6.2 Qualitative Evaluation

The prediction results for a few randomly selected samples from the validation dataset are visualized in Figs. 4.8a and 4.8b. The first row represents 5 frames out of 100 frames of one reference sample. The second row illustrates the prediction corresponding to the frames in the first row, and the last row represents the error in the corresponding predictions. The columns represent the time steps 1, 25, 50, 75 and 100 seconds. We visualized frames at intervals of 25 seconds to evaluate different ranges of dynamic stress prediction.

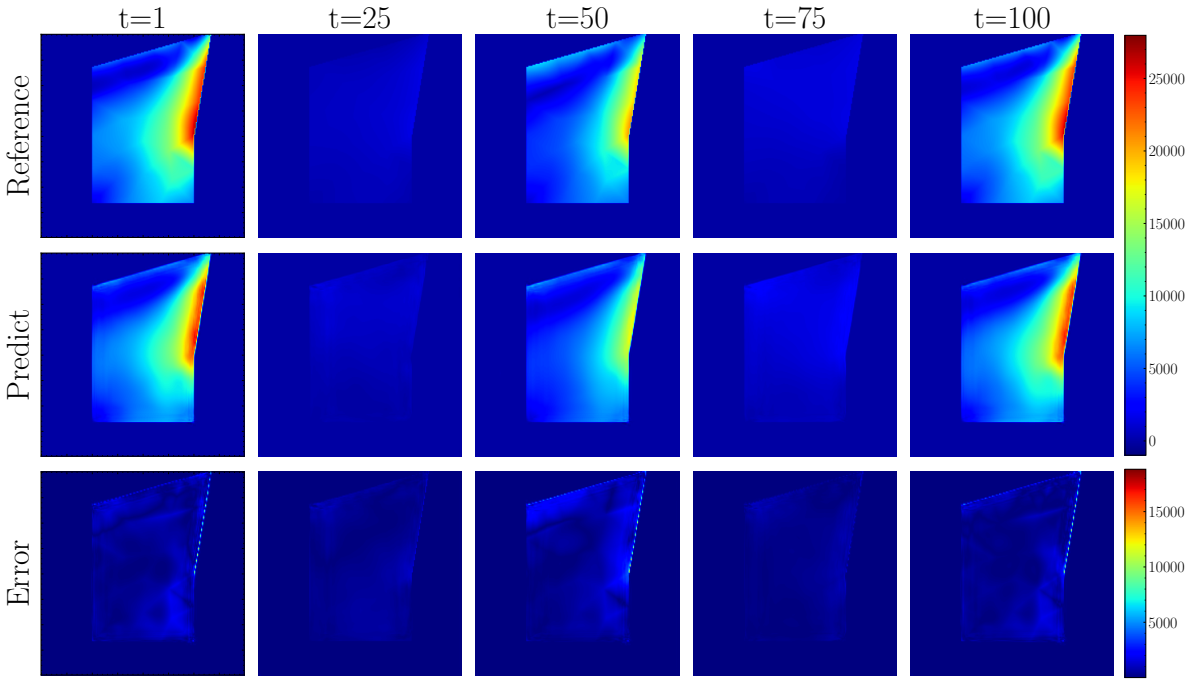
For visualization purposes, the references and predictions in Figs. 4.8a and 4.8b are scaled to the same range using the maximum and the minimum of each sample. The errors are scaled independently. As it can be seen in Fig. 4.8a, the predicted frames are quite similar to their corresponding references. Although the geometry contains sharp corners and edges, which are areas that are hard for CNN to reconstruct, our model can predict it. The errors, except for a small part of the first frame, are in an acceptable range which shows the prediction accuracy of our model. Fig. 4.8b shows another successful reconstruction. Comparing references with their corresponding predicted frames demonstrates that our Neuro-DynaStress model can capture both load variations and maximum stress values at the same time. Furthermore, these results demonstrate that our model can predict a dynamic stress distribution with a high variation of distributed stress.

Fig.4.9 shows a random failure sample. Despite the model's success in predicting most parts of the frames, it is not able to reconstruct high-stress concentrations at angles of 90 degrees. Since CNNs typically struggle in handling sharp edges, smoothening the sharp corners using Gaussian filters during data preprocessing may help the network to train better. Furthermore, as the loads in frames $t = 25$ and $t = 75$ are lower than in other frames, the prediction in those frames is acceptable.

It is also important that the predictions are temporally consistent. In order to qualitatively



(a)



(b)

Figure 4.8 Successful predicted dynamic stress distribution and their corresponding errors in different time sequences for two samples. The top row corresponds to reference frames and the middle row shows the predictions. The bottom row shows the absolute error between corresponding frames (Unit = MPa).

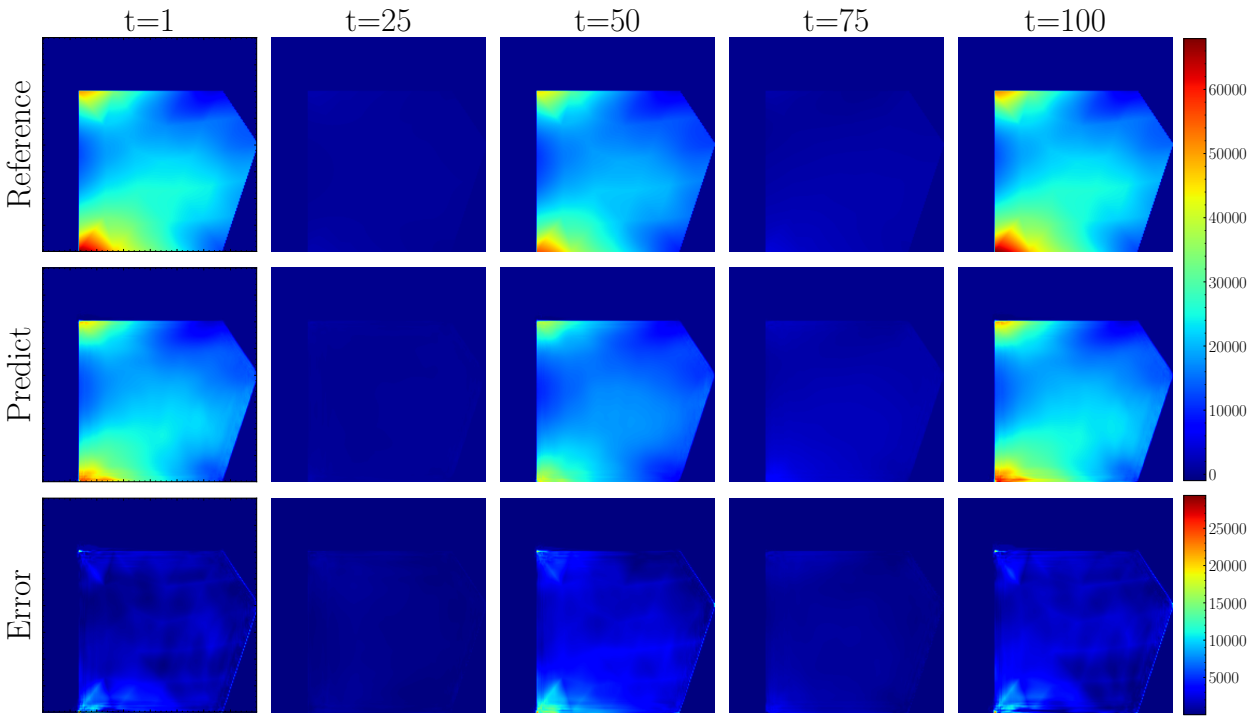
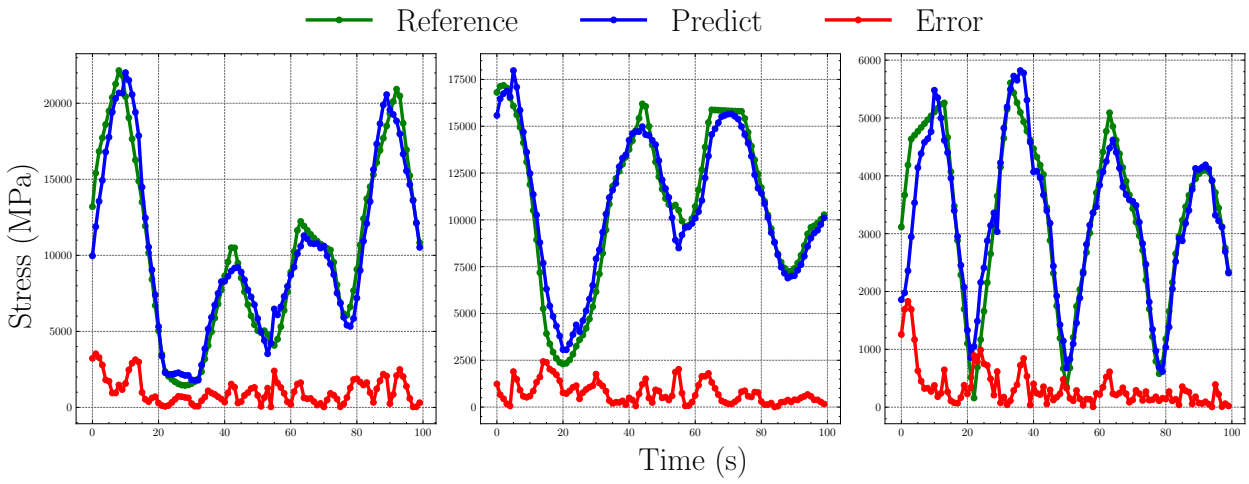


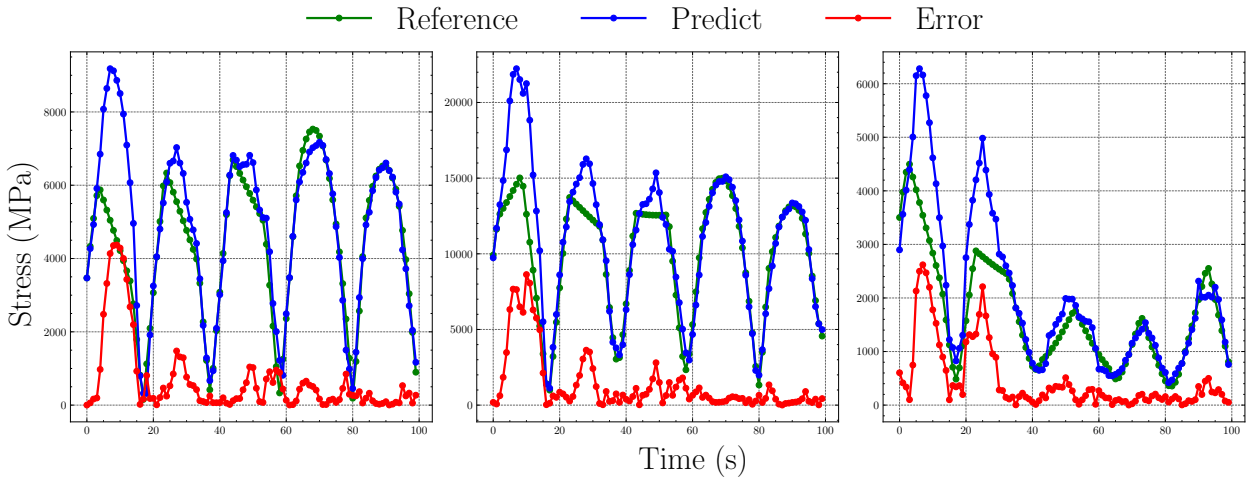
Figure 4.9 Failed predicted dynamic stress distribution and their corresponding errors in different time sequences (Unit = MPa).

demonstrate the temporal consistency of the proposed method, Fig. 4.10a shows a comparison of stress values across 100 frames for successful predictions in a randomly selected element. As can be seen, the references and the predicted distributions are almost identical in most time sequences, with errors close to zero, despite the stress varying widely with time. Fig. 4.10a illustrates how prediction fits with reference more closely when there is more temporal smoothness at peak points. For instance, a good match between prediction and reference can be seen in the rightmost graph in Fig. 4.10a, where the stress variation follows a smooth Gaussian distribution in the last peak. However, in the remaining graphs, the prediction has good correlation with the reference despite a lack of smoothness in most peak stress values. Moreover, based on the graphs in Fig. 4.10a, we can conclude that the model is better at predicting stress in valleys compared to peaks.

We have also illustrated some of the unsuccessful predictions in Fig. 4.10b to identify the limitations of our proposed model. It can be seen that in all graphs with non-Gaussian stress distributions, the model finds it difficult to capture the peak stress values accurately. However, in the first two graphs from the left in Fig. 4.10b, the predictions perfectly fit later peaks of the reference



(a)



(b)

Figure 4.10 Comparison of stress values across 100 frames for predictions, references, and errors in a randomly selected element. (a) Successful predictions (b) Unsuccessful predictions (Units = MPa-T).

since the stress values in the reference have Gaussian distributions at these points. Figs. 4.11a and 4.11b depict the MRPE of randomly selected samples across 100 frames and frames corresponding to the minimum and maximum MRPE. As can be seen for both samples, the minimum errors are around zero, with only a few frames exceeding the error by more than 2%.

4.6.3 Ablation Study

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

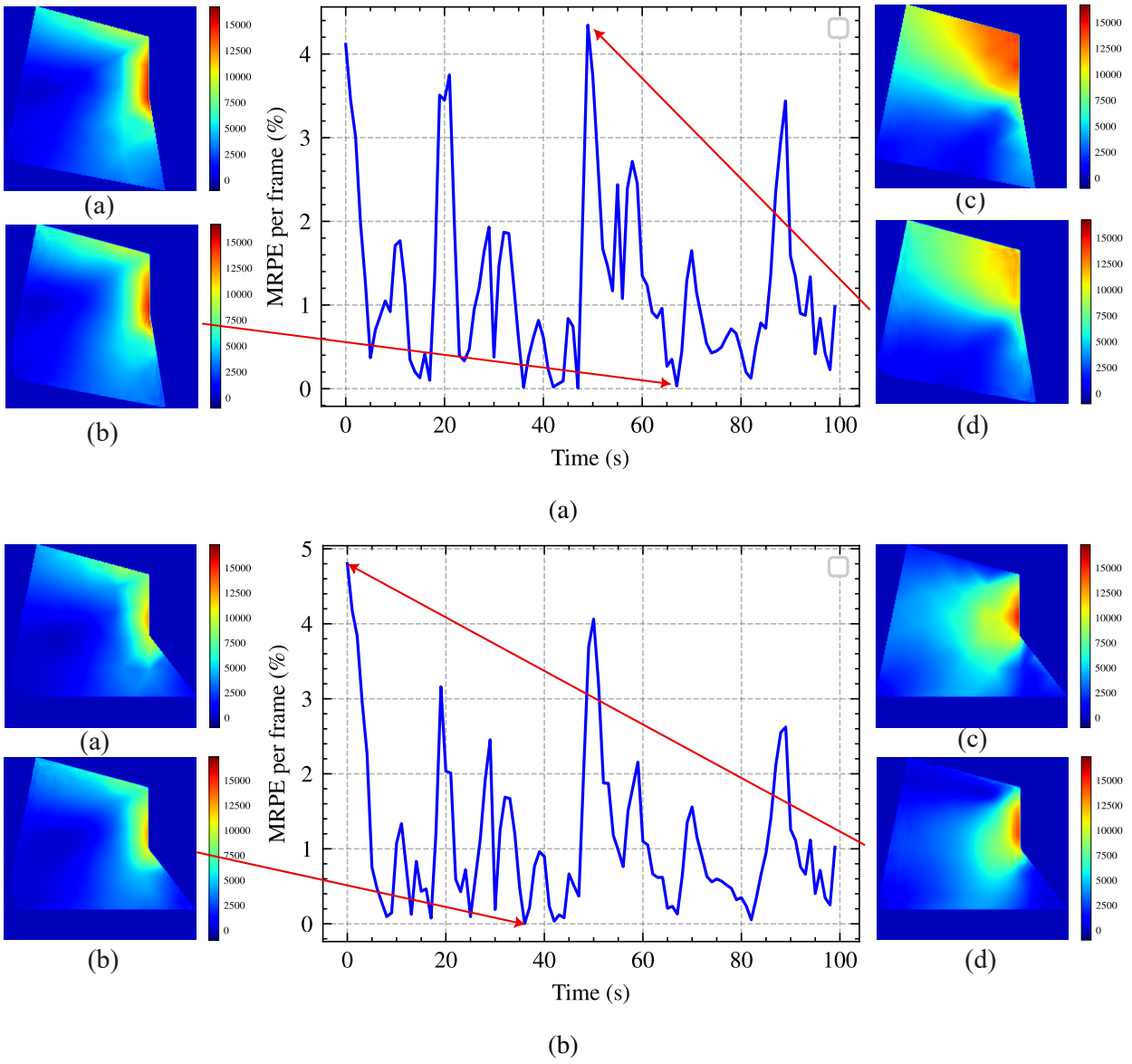


Figure 4.11 Relative errors across 100 frames in the randomly selected sample. Graphs in the center represent the MRPE per frame. (a) and (c) in each figure represent the reference; (b) and (d) refer to their corresponding predictions. Arrows refer to the MRPE of the presented frame (Units = MPa-T).

different elements in an input. 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 as shown in Table 4.3. The model with multi head self-attention is very similar to our architecture, except the LSTM modules in our model are replaced with self-attention modules. The details of the other models are represented

in Table 4.3. We will refer to our architecture as Neuro-DynaStress. The results are shown in Table 4.3, and the best results are highlighted in bold.

Table 4.3 Architecture comparison

Architecture for modeling temporal information				
	Multi-headed self-attention	LSTM	LSTM	LSTM
FaPN	✓	✓	✓	×
Skip connection	✓	✓	×	×
MRPE(%)	4.5	**2.3**	6.6	9.7

CHAPTER 5

PHYSICS INFORMED NEURAL NETWORK FOR DYNAMIC STRESS PREDICTION

5.1 Physics Informed Neural Network

Data collection and generation are common themes across various scientific fields, but data assimilation is far more challenging. While many ML models have shown some initial success and promise, the majority are unable to extract interpretable information and knowledge from this huge amount of data. Moreover, purely data-driven models may fit observations very well, but predictions may be physically inconsistent or implausible, owing to extrapolation or observational biases that may lead to inappropriate generalization. Furthermore, purely data-driven models may be accurate at fitting observations, but be physically incoherent or implausible in their predictions due to extrapolation bias or observational errors, which may lead to inappropriate generalization. Therefore, to integrate fundamental physical laws and domain knowledge, it is imperative that machine learning models learn about the rules governing physical behavior. Physical laws can provide a strong theoretical constraint and inductive bias, in addition to observational ones, that can serve as 'informative priors'. Consequently, it is necessary to implement physics-informed learning, where prior knowledge derived from observation, empirical, physical, or mathematical understanding of the world can enhance a learning algorithm's performance [57]. Presas et al [62] 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 [63, 64] 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 [65, 66]. Raissi et al. [67] 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 [68] introduced and illustrated the PINN approach. Vahab et al [69] 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 [70] proposed an approach to solving linear elasticity problems in composite plates and tubes using Physics Informed Neural Network. Chen et al [71] 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 [72] proposed an advanced PINN method based on the modified loss function for computational 2D and 3D solid mechanics. Jeong et al [73] 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 [74] 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 [75] proposed a PINN model for bending and free vibration analysis of three-dimensional functionally graded porous beams. Bazmara et al [76] 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 [77] presented a PINN model for temperature field predicting heat source layout. Xu et al [78] introduced a PINN model for predicting external loads of diverse engineering structures based on limited displacement monitoring points. Zheng et al [79] reconstructed the solution of the displacement field after damage to predict crack propagation using PINN. Yao et al. [80] proposed a physics-guided learning algorithm for predicting the mechanical response of materials and structures. Das et al. [81] proposed a data-driven physics-informed method for prognosis and applied it to predict cracking in a mortar cube specimen. Wang et al. [82] proposed a hybrid DL model that unifies representation learning and turbulence simulation techniques using physics-informed learning. Goswami et al. [83] proposed a physics-informed variational formulation of DeepONet for brittle fracture analysis. Raissi et al. [67] 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. Haghighat et al. [84] presented physics-informed neural networks to inversion and surrogate modeling in solid mechanics. Jin et al. [85] investigated the ability of PINNs to directly simulate incompressible flows, ranging from laminar to turbulent flows to turbulent channel flows. Li et al. [86] used the Fourier transform to develop a Fourier neural operator to model turbulent flows.

The recently introduced models [87, 88] 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 [64, 65, 89] 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 incorporate partial differential equations (PDEs) into the training loss function directly. 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 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 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 such as ours can help avoid catastrophic failures. In practice, the outputted stress maps from our models can be used by downstream applications for detecting anomalies such as cracks in the plates. In other words, it can act as a precursor to existing vision-based systems.

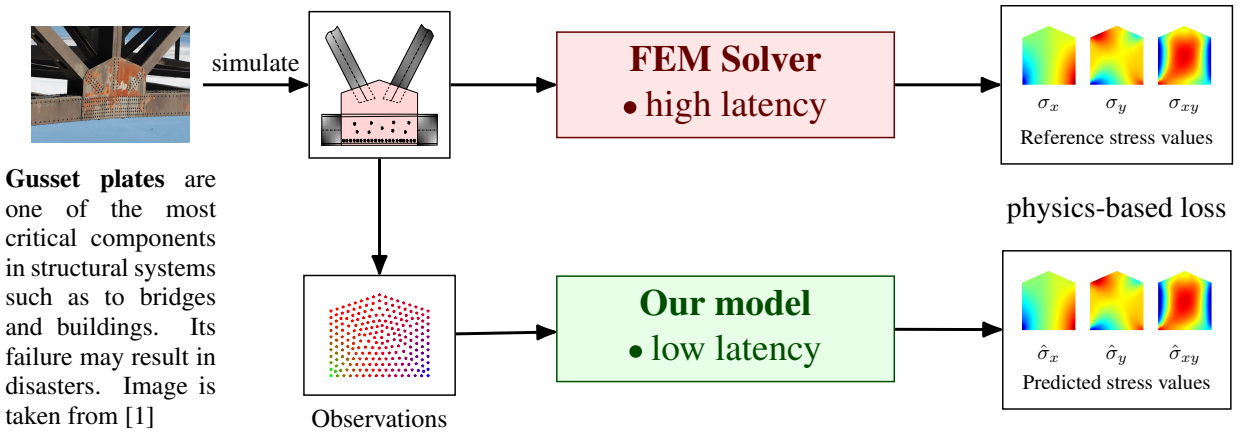


Figure 5.1 **Overview:** Unlike FEM, PINN-Stress is computationally efficient, facilitates real-time analysis. PINN-Stress use a governing equation behind the equation of motion as a soft constraint in the loss function to enforce the loss to minimize. The points with different colors in observations correspond to the same nodes in the gusset plate. Gusset plate image is taken from [90].

An overview of our approach is shown in Fig. 5.1. 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. Loss function in PINN-Stress uses traditional MAE loss for training. PINN-Stress uses the physics-informed loss function described in Section 3.1. For real-life use, our models require input from sensors placed on the plates. 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. Introducing novel spatiotemporal multiplexing to physics-informed learning and showing its utility in dynamic stress prediction;

4. 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;

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.

5.2 Background

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:

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + b_x - \rho a_x = 0 \tag{5.1}$$

$$\frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{xy}}{\partial x} + b_y - \rho a_y = 0 \tag{5.2}$$

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.

5.2.1 von Mises equation

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.

$$\sigma_{vm} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 - \sigma_{xx}\sigma_{yy} + 3\sigma_{yy}^2} \tag{5.3}$$

5.3 Method

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

5.3.1 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 [34] or self-attention [56]. 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. We would like to point out that the idea of multiplexing is not novel in deep learning literature [59, 91]. Our contribution is that we are the first to introduce 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. 5.2

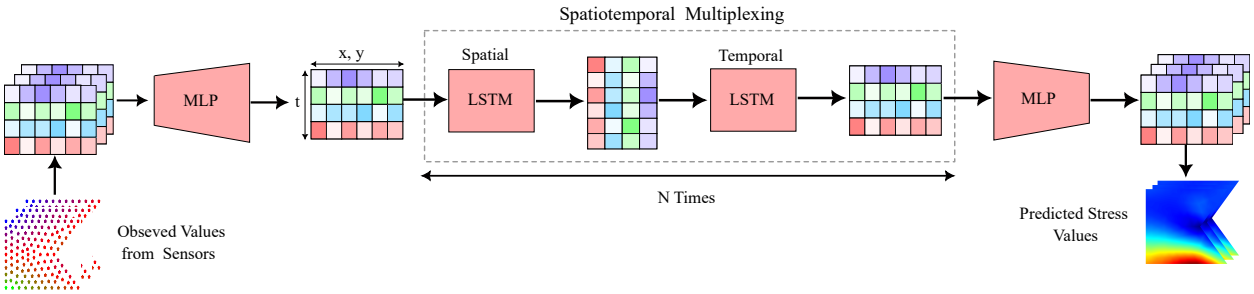


Figure 5.2 **Model architecture:** We introduce the novel spatiotemporal multiplexing (STM) to physics-informed learning in order to learn both spatial and temporal information in the data. Our architecture is lightweight and hence gives a real-time performance.

5.3.2 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. 5.1 and 5.2. 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:

$$\mathcal{L} = w_{\text{data}}\mathcal{L}_{\text{data}} + w_{\text{PDE}}\mathcal{L}_{\text{PDE}} + w_{\text{bc}}\mathcal{L}_{\text{bc}} \tag{5.4}$$

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 by calculating the mean absolute error between the LHS and the 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:

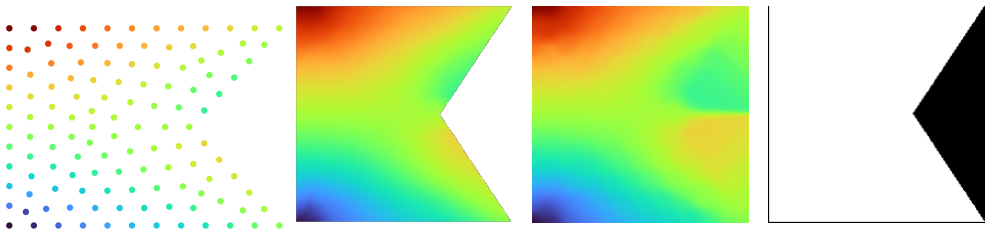
$$\sigma(x, y, t = 0) = 0 \tag{5.5}$$

$$\sigma(x, y, (t_0 \ldots t_n)) = \sigma \tag{5.6}$$

Equations 5.5 and 5.6 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.

5.3.3 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 to calculate 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. 5.3. 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. 5.3c, the grid is accurate within the mesh region. Now, we can estimate the gradients for the stress outputs from these grids.



(a) (b) (c) (d)

Figure 5.3 **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.

5.4 Experiments and Results

5.4.1 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 [41, 42, 43]. The boundary conditions and time-history load cases are

considered to simulate similar conditions in common gusset plate structures under external loading.

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. 5.4, 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. 5.5, are considered to be made of homogeneous and isotropic linear elastic materials. Various geometries are generated by changing the position of each node in horizontal and vertical directions, as shown in Fig. 5.5, which leads to 1024 unique pentagons. The material properties remain unchanged and isotropic for all samples.

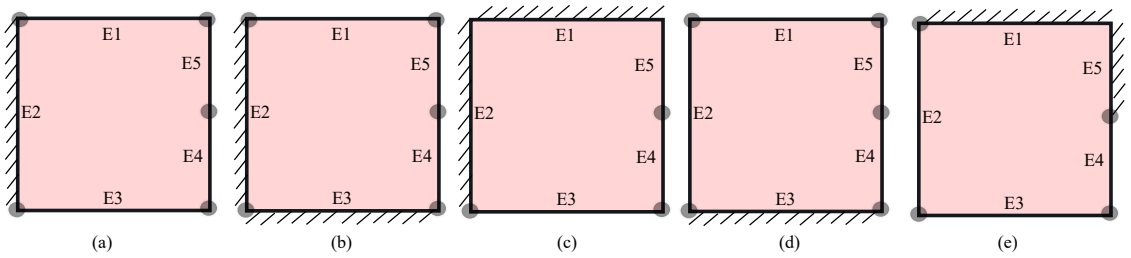


Figure 5.4 Different types of boundary conditions for initializing population.

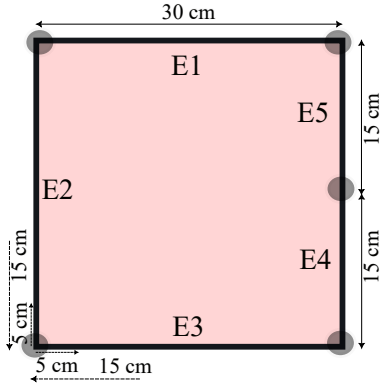
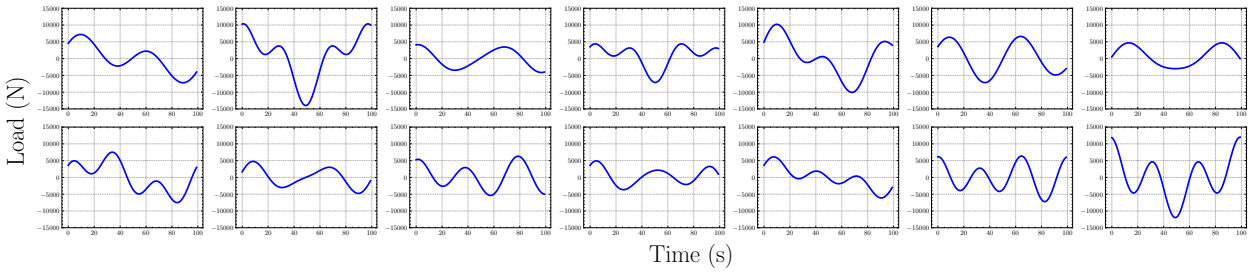


Figure 5.5 Basic schematic topology for initializing the steel plate geometries.

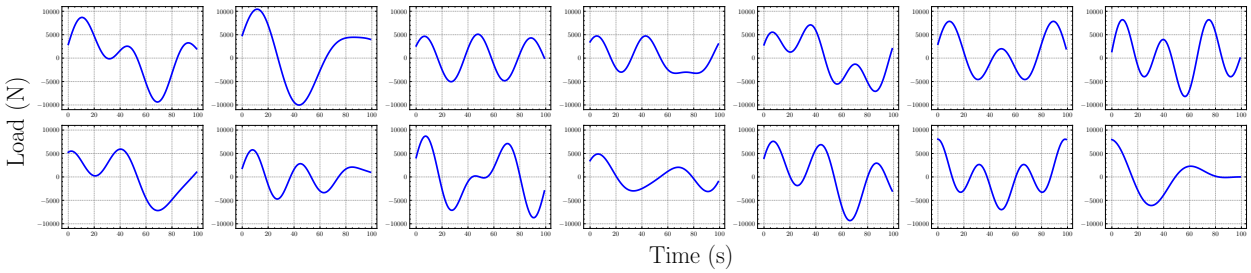
Time histories consist of 100 time-steps generated with random sine and cosine frequencies. The frequencies range between 1 and 3 Hz, with amplitudes ranging from 2 to 10 kN at intervals of 2 kN. All time histories in horizontal and vertical directions are shown in Fig. 5.6. Each time series

Table 5.1 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



(a)



(b)

Figure 5.6 Various load sequences in (a) horizontal and (b) vertical directions.

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 5.1.

5.4.1.1 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. 5.7 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 5.1.

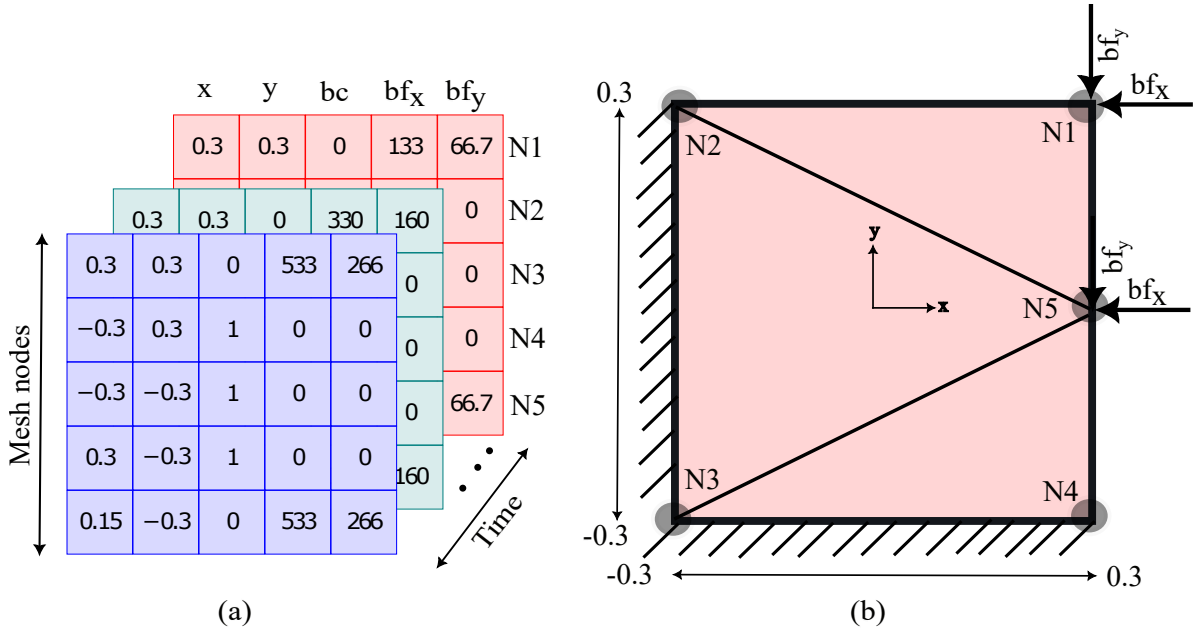


Figure 5.7 Construction of input matrix (Unit: m, N).

5.4.1.2 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 nodes, the size of the output matrix is 200×100. Each of the three outputs is normalized separately between -1 and 1 to ensure faster convergence. The input and the output representations of the model are shown in Fig. 5.8.

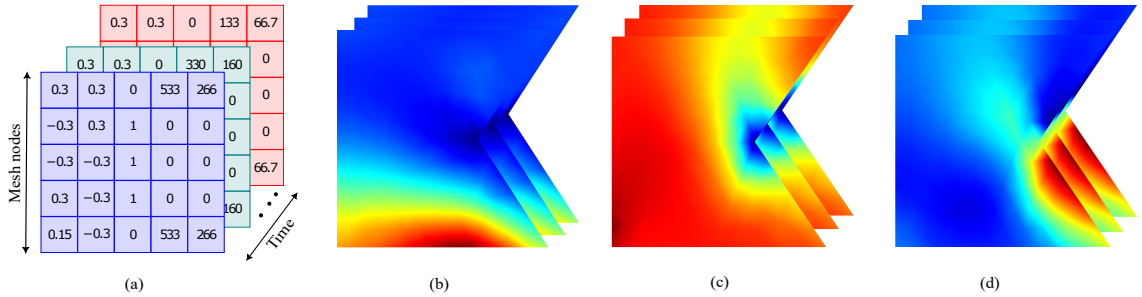


Figure 5.8 Input and output representation for normal and shear stress distribution prediction: (a) Input matrix, (b) Output (σ_{xx}), (c) Output (σ_{yy}), (d) Output (σ_{xy}).

5.4.2 Metrics

We use Mean Absolute Error (MAE), defined in Eq. 5.7 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.

$$\text{MAE} = \frac{1}{NT} \sum_{N,T}^{n,t} \left| S(n,t) - \hat{S}(n,t) \right| \tag{5.7}$$

$$\text{MRPE} = \frac{\text{MAE}}{\max(|S(n,t)|, |\hat{S}(n,t)|)} \times 100 \tag{5.8}$$

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.

5.4.3 Implementation

We implemented our model using PyTorch [61] and PyTorch Lightning. AdamW optimizer [59] was used with an initial learning rate of 10^{-3}. We found that a batch size of 10 gives the best results. 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 which satisfies the real-time requirement. The most

powerful FE solvers take between 10 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 consider the minimum time for all processes of modeling geometry, meshing, and analysis of one sample in the FE solver to be about 10 minutes. MATLAB PDE solver does not use GPU acceleration. Therefore, NeuroStress and PINN-Stress are about 6×10^5 faster than MATLAB PDE solver.

5.4.4 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. 5.9 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.

Table 5.2 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.

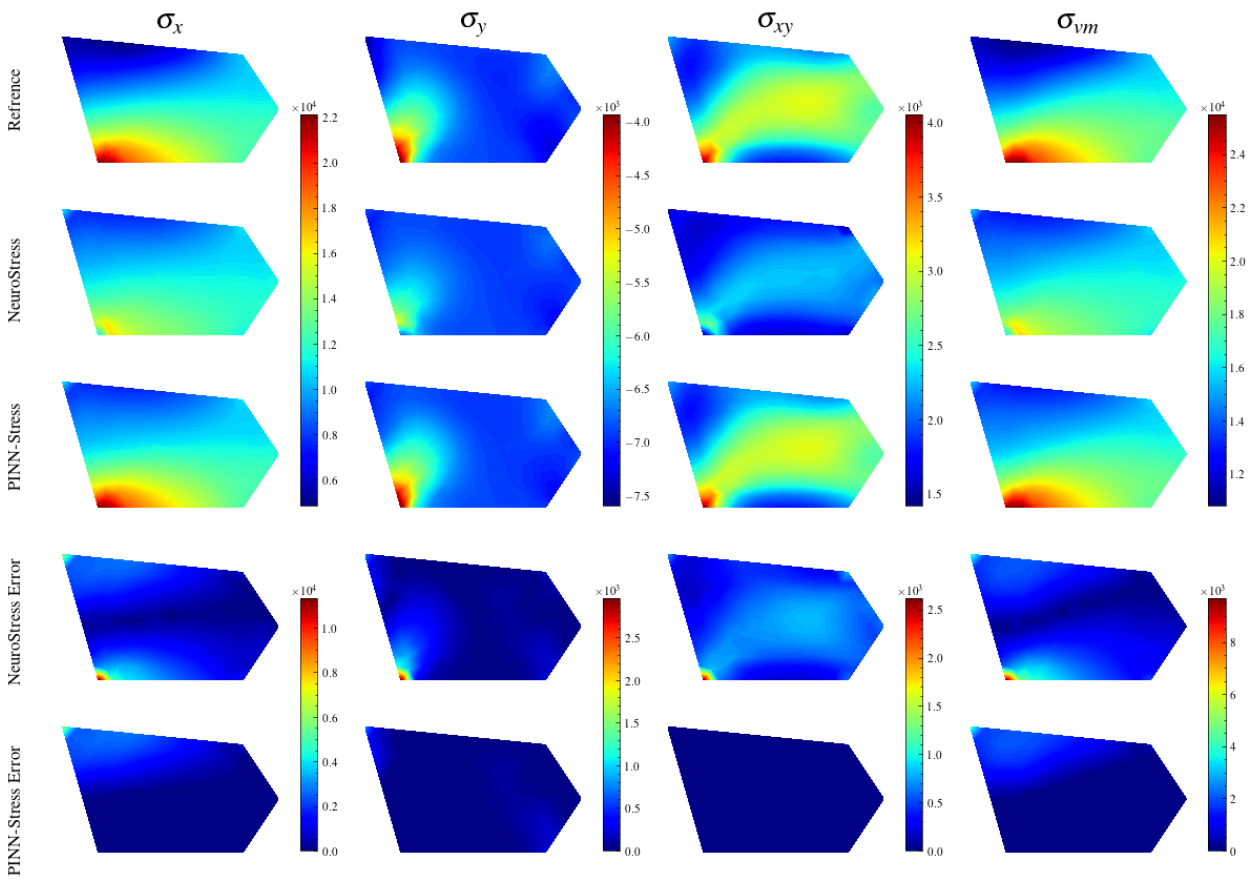


Figure 5.9 Comparison of NeuroStress and PINN-Stress predictions for σ_{xx}, σ_{yy}, σ_{xy} and σ_{vm} (Unit: MPa).

5.5 Ablation Studies

5.5.1 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 the 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, the training set will consist of 614 geometries out of 1024, and validation and test sets will contain the remaining (205 each). 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. Figs 5.10 and 5.11 demonstrate the generalization capability of PINN-Stress and

NeuroStress to unseen load sequences and geometries, respectively. 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.

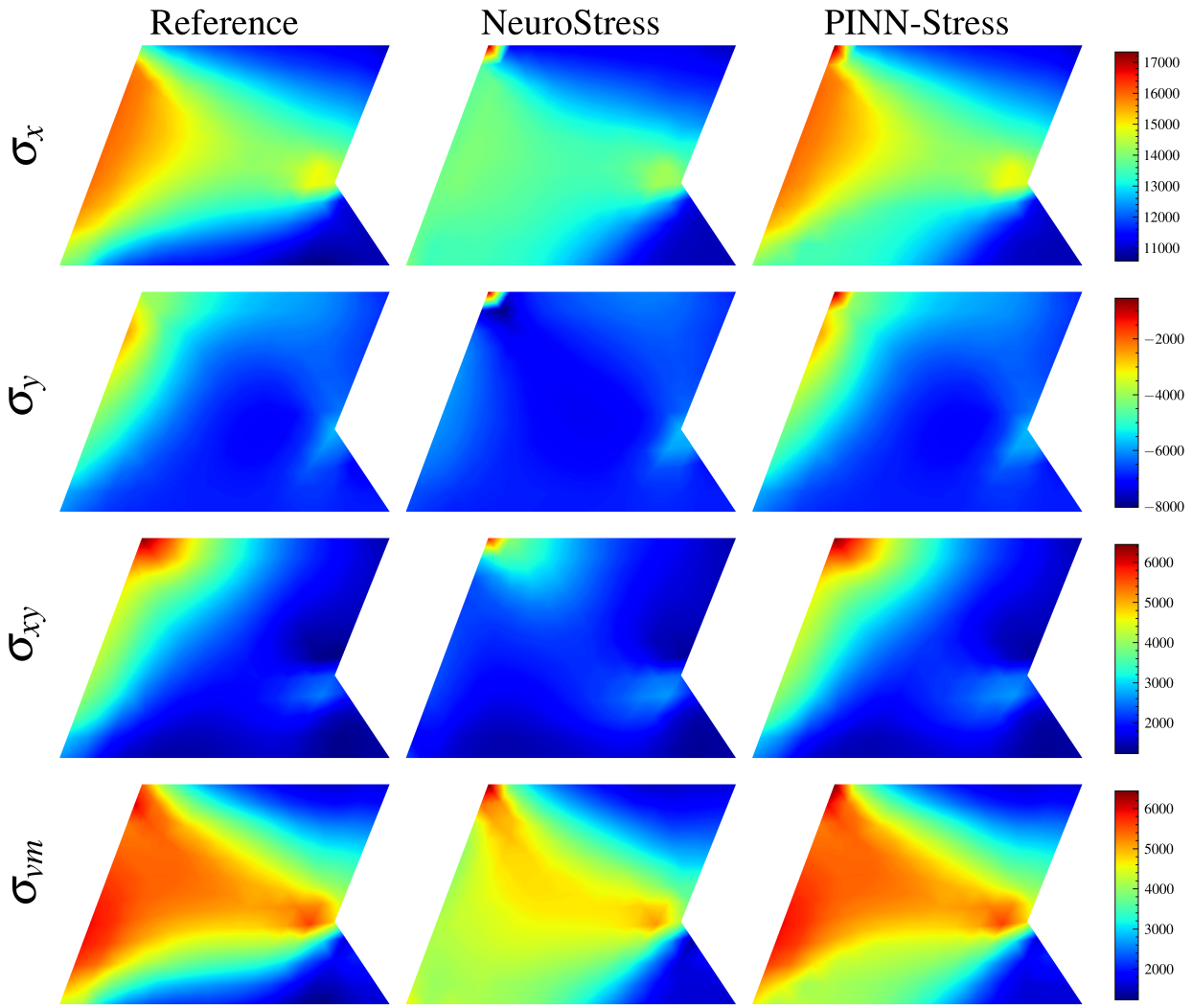


Figure 5.10 **Predicting dynamic stress distribution for diverse load sequences:** Augmenting our novel architecture with a physics-based loss can induce generalization capabilities while still remaining real-time (Unit: MPa). The overview of our method is given in Fig. 5.1.

Fig 5.12 shows the error of each frame for a random spatial node across all time frames for unseen load sequences and structural geometries. As it can be seen, in both figures, the errors in PINN-Stress are less than NeuroStress, especially in extreme peaks, which demonstrates the ability of PINN-Stress to predict the maximum stress values. We have also compared the generalization capability of PINN-Stress and NeuroStress over unseen load sequences and geometries in a single

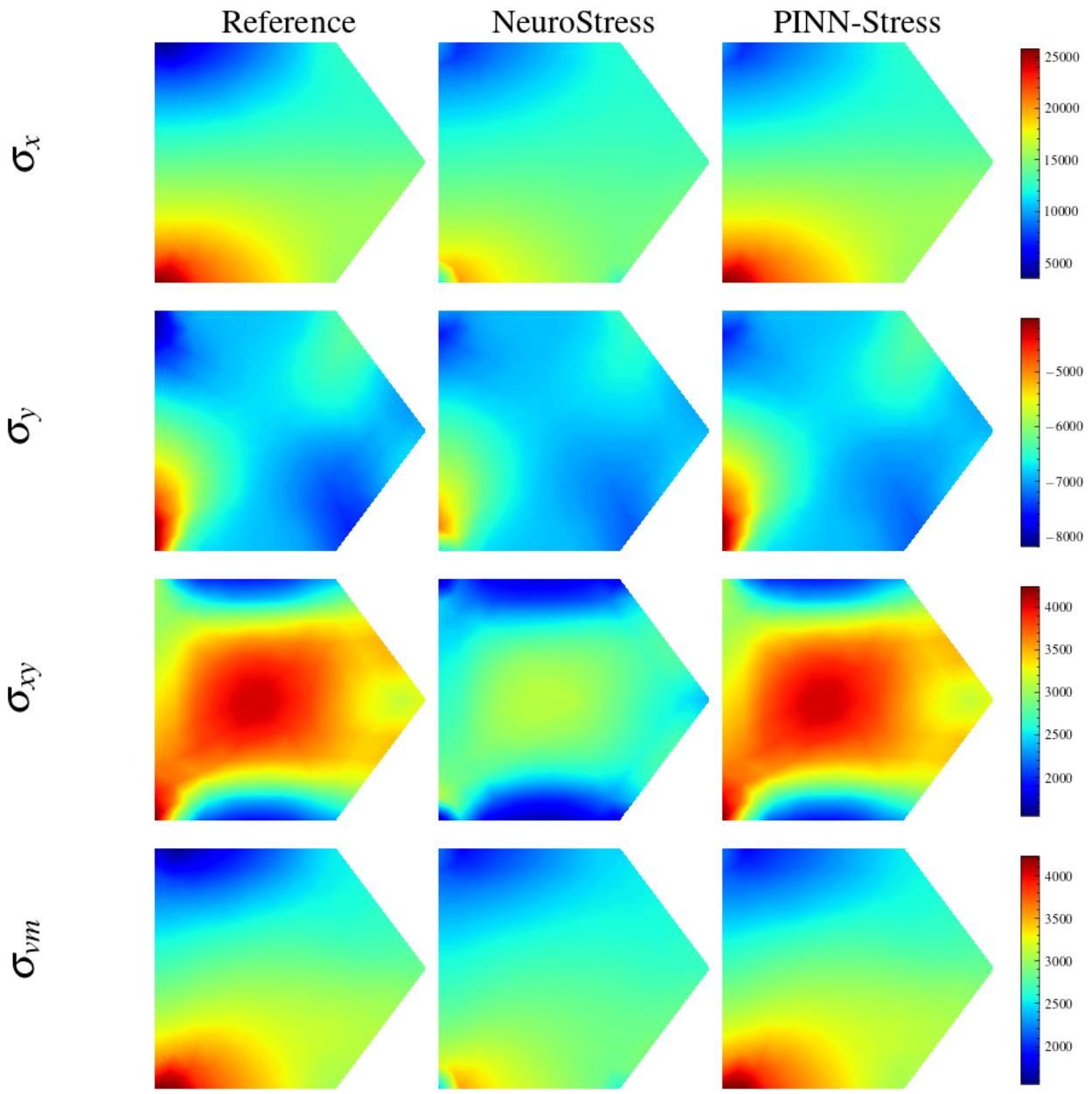


Figure 5.11 Predicting dynamic stress distribution for diverse geometries (Unit: MPa).

spatial node across all time frames in Figs 5.13 and 5.14. Figs 5.13 and 5.14 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. Table 5.2 shows the data split for each experiment and the corresponding results. 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. The results were similar.

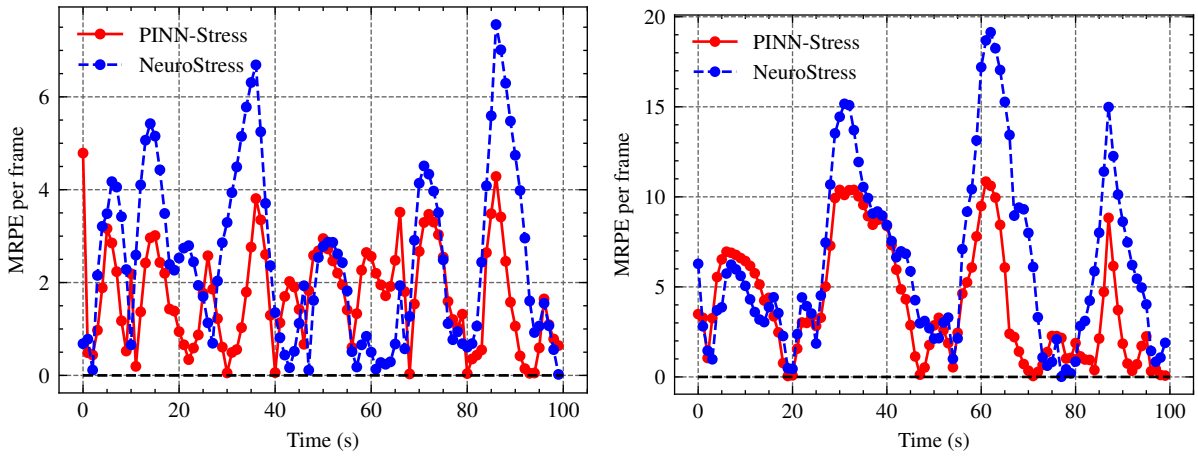


Figure 5.12 Comparison of NeuroStress and PINN-Stress errors for σ_{vm} across 100 frames for a random spatial node in a sample. (a) unseen load sequences and (b) unseen geometries.

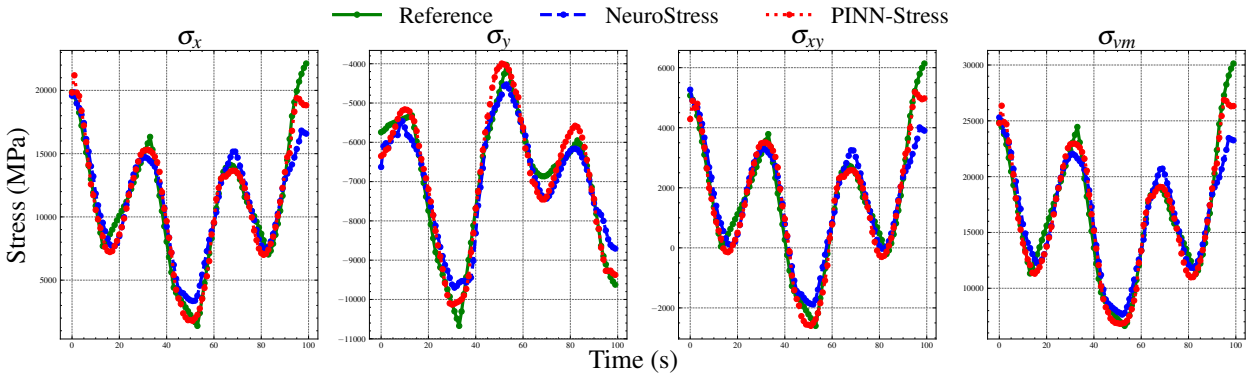


Figure 5.13 Comparison of NeuroStress and PINN-Stress predictions for σ_{xx}, σ_{yy}, σ_{xy} and σ_{vm} across 100 frames for a sample with unseen load sequences.

5.5.2 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,

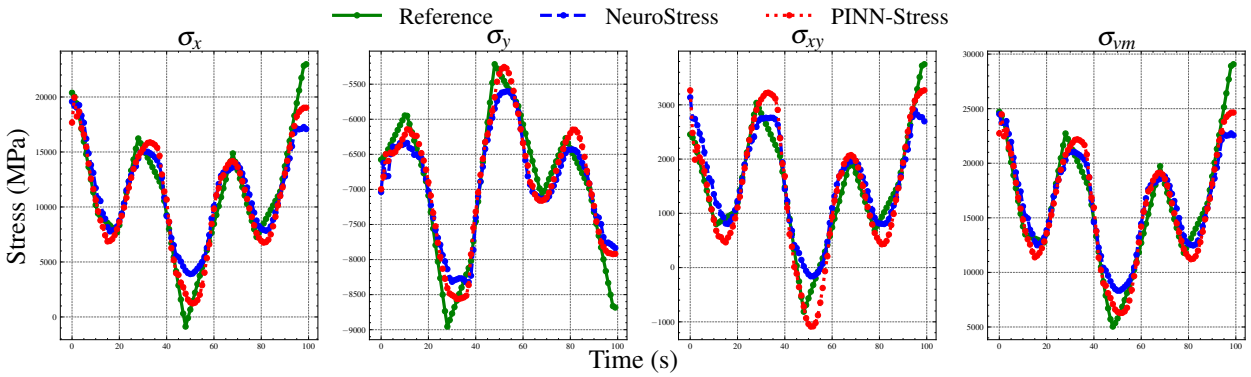


Figure 5.14 Comparison of NeuroStress and PINN-Stress predictions for σ_{xx}, σ_{yy}, σ_{xy} and σ_{vm} across 100 frames for a sample with unseen geometries.

we are the first to introduce such a design into PINNs to the best of our knowledge. We train our dataset using both LSTM and self-attention for different dataset sizes to ensure which of them has a better performance based on our dataset. As illustrated in Fig 5.15, LSTM consistently demonstrated better performance compared to self-attention, with lower error rates. This suggests that LSTM is more suitable for our dataset compared to self-attention. 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 also compare our architecture against other baseline architectures.

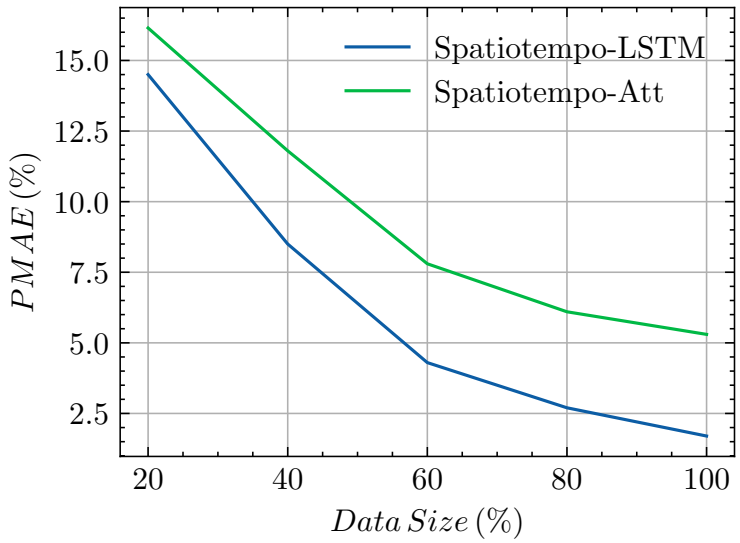


Figure 5.15 comparison of LSTM and self-attention for different training data size.

We compare against three architectures: **Spatiotempo-Att**, **Tempo-LSTM**, **Spatio-MLP**.

Table 5.3 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**

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 for 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 5.3, and the best results are highlighted in bold.

CHAPTER 6

SUMMARY AND CONCLUSION

This study presents a framework for stress distribution prediction in structural components utilizing deep learning techniques. Our models can predict both static and dynamic stress distribution.

In the first project, we used end-to-end DL techniques. We developed a CNN to alleviate the need for finite element methods for the prediction of high-resolution stress distributions in loaded steel plates. The CNN was designed and trained to use the geometry, boundary conditions, and load as input, providing high-resolution stress contours as the output. We used the PDE toolbox of MATLAB to generate the output data for training, containing 104,448 FEM samples. We trained and evaluated different models to find the model with the best performance. The best model can predict the stress distributions with a mean absolute error of 0.9% and a maximum stress error of 0.46% in the von Mises stress distribution. The effects of dataset size on the model performance were also studied. Training the network with just 10% of the dataset achieved a mean error of 1.85%, which can be considered acceptable in specific engineering applications. Moreover, we evaluated the effect of dataset size on the Gaussian distribution of mean and maximum stress errors. Increasing the data size decreased the standard deviation of mean error. The standard deviation of maximum stress error also decreased with increase of the number of samples. Furthermore, the Gaussian distributions of mean and maximum stress errors demonstrated that a greater quantity of data induced less standard deviation in PMAE and PPAE.

In the second project we develop a convolutional neural network (CNN) augmented with the custom loss function which is inspired from stress concentration physics equation to predict high-resolution von Mises stress distribution in the specific domain of damaged steel plates. The proposed network learns to predict the stress distribution given the damaged geometries, load, and boundary conditions as input and high-resolution stress contours as the output. The dataset is composed of 61,440 unique and complex cases of various geometries, boundary conditions, and loads. The PDE toolbox of MATLAB was used to generate the output data for training as FEA samples. We

also build a CNN model using torch.nn.MSELoss function to see how much our proposed custom loss function is efficient. The CNN network achieves high accuracy in both custom loss and MSE models, under multiple metrics, in the evaluations of stress distribution datasets. The custom loss model outperforms the MSE model in terms of peak stress value predictions, in addition to accurately localizing damages and capturing stress concentration around crack tips, which is not possible with other ML methods. The custom loss trained DL model which trained with 49152 FEA samples can be used for future predictions of stress distributions of damaged steel plates of 12888 FEA samples. The custom loss trained DL model has a mean absolute error of 0.22% and a maximum stress error of 1.5% in the von Mises stress distribution.

In the third project We propose Neuro-DynaStress model equipped with Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) to predict the entire sequence of dynamic stress distribution. The model was designed and trained to use the geometry, boundary conditions, and the sequence of loads as input and predicts the sequence of high-resolution dynamic stress contours. The convolutional components are used to extract spatial features, and the LSTM captures the temporal dependence between the frames. Feature alignment modules are used to improve the training and performance of our model. The model is trained using synthetic data generated using the PDE toolbox in MATLAB. Neuro-DynaStress can predict dynamic stress distribution with a mean relative percentage error of 2.3%, which is considered an acceptable error rate in engineering communities.

In the fourth Project We propose NeuroStress and PINN-Stress, two models for dynamic stress prediction based on a novel architecture, with the latter augmented with a 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 are 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.

CHAPTER 7

FUTURE WORKS

Develop a deep learning model for 3-D geometries which can predict stress distribution in more real and complex samples. Convolutional neural networks (CNNs) have been widely used for image-based deep learning tasks, and they can also be used for 3-D geometries. However, 3-D CNNs can be computationally expensive, therefore, other architectures, such as graph neural networks (GNNs), or using matrix-based approaches like the way we embedded our input and output dataset into the matrix in the last chapter may be more suitable for 3-D geometries. This can be extremely useful in many fields, such as aerospace engineering, civil engineering, and biomechanics, where accurate predictions of stress distribution are critical for ensuring the safety and reliability of structures and systems.

Create a deep learning model to predict vibration modes and frequencies of structural components. Engineers need to do modal analysis before most dynamic analysis to calculate vibration modes and natural frequencies of the model. This helps them to have more intuition about the global and local behavior of their model. A deep learning model for predicting vibration modes and natural frequencies of a model can be more efficient than traditional me thods. Once the model is trained, it can provide predictions quickly and efficiently, without the need for lengthy simulations or computations.

Propose a model to predict temperature distribution, radial stress, and hoop stress for real-world scenarios such as disc brakes and pipes. The heat is generated by the friction in disk brakes and then dissipated through the disc and surrounding components. Since the lifetime of disc brakes is important for automotive companies' prediction of temperature distribution can increase disc brakes' lifetime. This can be integrated with a digital twin model which will provide real-time monitoring of disk brake temperature.

Predicting crack propagation using sequential algorithms such as LSTM and transformers augmented with PINN. This would be better to perform with mesh-free methods which have been

applied to a variety of engineering problems involving complex geometries, large deformations, or discontinuities. The mesh-free method since does not require mesh updating for modeling the propagation of cracks could be an accurate and efficient approach for predicting crack propagation.

BIBLIOGRAPHY

[1] N. Umetani, "Exploring generative 3d shapes using autoencoder networks," in *SIGGRAPH Asia 2017 Technical Briefs*, pp. 1–4, 2017.

[2] Y. Yu, T. Hur, J. Jung, and I. G. Jang, "Deep learning for determining a near-optimal topological design without any iteration," *Structural and Multidisciplinary Optimization*, vol. 59, no. 3, pp. 787–799, 2019.

[3] A. B. Farimani, J. Gomes, and V. S. Pande, "Deep learning the physics of transport phenomena," *ArXiv Preprint ArXiv:1709.02432*, 2017.

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

[5] G. B. Goh, N. O. Hodas, and A. Vishnu, "Deep learning for computational chemistry," *Journal of Computational Chemistry*, vol. 38, no. 16, pp. 1291–1307, 2017.

[6] A. Mardt, L. Pasquali, H. Wu, and F. Noé, "Vampnets for deep learning of molecular kinetics," *Nature Communications*, vol. 9, no. 1, pp. 1–11, 2018.

[7] A. Mohammadi Bayazidi, G.-G. Wang, H. Bolandi, A. H. Alavi, and A. H. Gandomi, "Multigene genetic programming for estimation of elastic modulus of concrete," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[8] M. Sarveghadi, A. H. Gandomi, H. Bolandi, and A. H. Alavi, "Development of prediction models for shear strength of sfrcb using a machine learning approach," *Neural Computing and Applications*, vol. 31, no. 7, pp. 2085–2094, 2019.

[9] S. M. Mousavi, P. Aminian, A. H. Gandomi, A. H. Alavi, and H. Bolandi, "A new predictive model for compressive strength of hpc using gene expression programming," *Advances in Engineering Software*, vol. 45, no. 1, pp. 105–114, 2012.

[10] H. Bolandi, W. Banzhaf, N. Lajnef, K. Barri, and A. H. Alavi, "An intelligent model for the prediction of bond strength of frp bars in concrete: A soft computing approach," *Technologies*, vol. 7, no. 2, p. 42, 2019.

[11] M. J. Atalla and D. J. Inman, "On model updating using neural networks," *Mechanical Systems and Signal Processing*, vol. 12, no. 1, pp. 135–161, 1998.

[12] R. I. Levin and N. Lieven, "Dynamic finite element model updating using neural networks," *Journal of Sound and Vibration*, vol. 210, no. 5, pp. 593–607, 1998.

[13] Z. Fan, Y. Wu, J. Lu, and W. Li, "Automatic pavement crack detection based on structured prediction with the convolutional neural network," *ArXiv Preprint ArXiv:1802.02208*, 2018.

[14] C. V. Dung *et al.*, "Autonomous concrete crack detection using deep fully convolutional neural network," *Automation in Construction*, vol. 99, pp. 52–58, 2019.

[15] A. Javadi, T. Tan, and M. Zhang, "Neural Network for Constitutive Modeling in Finite Element Analysis," *Computer Assisted Mechanics and Engineering Sciences*, vol. 10, no. 4, pp. 523–530, 2003.

[16] A. Oishi and G. Yagawa, "Computational mechanics enhanced by deep learning," *Computer Methods in Applied Mechanics and Engineering*, vol. 327, pp. 327–351, 2017.

[17] A. Madani, A. Bakhaty, J. Kim, Y. Mubarak, and M. R. Mofrad, "Bridging finite element and machine learning modeling: stress prediction of arterial walls in atherosclerosis," *Journal of biomechanical engineering*, vol. 141, no. 8, 2019.

[18] L. Liang, M. Liu, C. Martin, and W. Sun, "A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis," *Journal of The Royal Society Interface*, vol. 15, no. 138, p. 20170844, 2018.

[19] N. S. Gulgec, M. Takáč, and S. N. Pakzad, "Convolutional neural network approach for robust structural damage detection and localization," *Journal of Computing in Civil Engineering*, vol. 33, no. 3, p. 04019005, 2019.

[20] C. Modarres, N. Astorga, E. L. Droguett, and V. Meruane, "Convolutional neural networks for automated damage recognition and damage type identification," *Structural Control and Health Monitoring*, vol. 25, no. 10, p. e2230, 2018.

[21] Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017.

[22] D. T. Do, J. Lee, and H. Nguyen-Xuan, "Fast evaluation of crack growth path using time series forecasting," *Engineering Fracture Mechanics*, vol. 218, p. 106567, 2019.

[23] T. T. Truong, D. Dinh-Cong, J. Lee, and T. Nguyen-Thoi, "An effective deep feedforward neural networks (dfnn) method for damage identification of truss structures using noisy incomplete modal data," *Journal of Building Engineering*, vol. 30, p. 101244, 2020.

[24] Q. X. Lieu, K. T. Nguyen, K. D. Dang, S. Lee, J. Kang, and J. Lee, "An adaptive surrogate model to structural reliability analysis using deep neural network," *Expert Systems with Applications*, vol. 189, p. 116104, 2022.

[25] X. Zhuang, H. Guo, N. Alajlan, H. Zhu, and T. Rabczuk, "Deep autoencoder based energy method for the bending, vibration, and buckling analysis of kirchhoff plates with transfer learning," *European Journal of Mechanics-A/Solids*, vol. 87, p. 104225, 2021.

[26] E. Samaniego, C. Anitescu, S. Goswami, V. M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk, "An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications," *Computer Methods in Applied Mechanics and Engineering*, vol. 362, p. 112790, 2020.

[27] J. Berg and K. Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries," *Neurocomputing*, vol. 317, pp. 28–41, 2018.

[28] V.-H. Truong, Q.-V. Vu, H.-T. Thai, and M.-H. Ha, "A robust method for safety evaluation of steel trusses using gradient tree boosting algorithm," *Advances in Engineering Software*, vol. 147, p. 102825, 2020.

[29] A. Khadilkar, J. Wang, and R. Rai, "Deep learning–based stress prediction for bottom-up sla 3d printing process," *The International Journal of Advanced Manufacturing Technology*, vol. 102, no. 5, pp. 2555–2569, 2019.

[30] Z. Nie, H. Jiang, and L. B. Kara, "Stress field prediction in cantilevered structures using convolutional neural networks," *Journal of Computing and Information Science in Engineering*, vol. 20, no. 1, p. 011002, 2020.

[31] H. Jiang, Z. Nie, R. Yeo, A. B. Farimani, and L. B. Kara, "Stressgan: A generative deep learning model for two-dimensional stress distribution prediction," *Journal of Applied Mechanics*, vol. 88, no. 5, 2021.

[32] D. Yinfeng, L. Yingmin, L. Ming, and X. Mingkui, "Nonlinear structural response prediction based on support vector machines," *Journal of Sound and Vibration*, vol. 311, no. 3-5, pp. 886–897, 2008.

[33] R.-T. Wu and M. R. Jahanshahi, "Deep convolutional neural network for structural dynamic response estimation and system identification," *Journal of Engineering Mechanics*, vol. 145, no. 1, p. 04018125, 2019.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[36] X. Fang, H. Li, S.-r. Zhang, X.-h. Wang, C. Wang, and X.-c. Luo, "A combined finite element and deep learning network for structural dynamic response estimation on concrete gravity dam subjected to blast loads," *Defence Technology*, 2022.

[37] C. P. Kohar, L. Greve, T. K. Eller, D. S. Connolly, and K. Inal, "A machine learning framework for accelerating the design process using cae simulations: An application to finite element analysis in structural crashworthiness," *Computer Methods in Applied Mechanics and Engineering*, vol. 385, p. 114008, 2021.

[38] M. Schwarzer, B. Rogan, Y. Ruan, Z. Song, D. Y. Lee, A. G. Percus, V. T. Chau, B. A. Moore, E. Rougier, H. S. Viswanathan, *et al.*, "Learning to fail: Predicting fracture evolution in brittle material models using recurrent graph convolutional neural networks," *Computational Materials Science*, vol. 162, pp. 322–332, 2019.

[39] M. Lazzara, M. Chevalier, M. Colombo, J. G. Garcia, C. Lapeyre, and O. Teste, "Surrogate modelling for an aircraft dynamic landing loads simulation using an lstm autoencoder-based dimensionality reduction approach," *Aerospace Science and Technology*, vol. 126, p. 107629, 2022.

[40] M. Jahanbakht, W. Xiang, and M. R. Azghadi, "Sediment prediction in the great barrier reef using vision transformer with finite element analysis," *Neural Networks*, vol. 152, pp. 311–321, 2022.

[41] S. M. Zahrai and M. Heidarzadeh, "Destructive effects of the 2003 bam earthquake on structures," *Asian Journal of Civil Engineering (Building and Housing)*, 2007.

[42] S. M. Zahrai and H. Bolandi, "Towards lateral performance of cbf with unwanted eccentric connection: A finite element modeling approach," *KSCE Journal of Civil Engineering*, vol. 18, no. 5, pp. 1421–1428, 2014.

[43] S. Zahrai and H. Bolandi, "Numerical study on the impact of out-of-plane eccentricity on lateral behavior of concentrically braced frames," *International Journal of Steel Structures*, vol. 19, no. 2, pp. 341–350, 2019.

[44] H. Bolandi and S. Zahrai, "Influence of in-plane eccentricity in connection of brace members to columns and beams on performance of steel frames," *Journal of Civil Engineering (Journal of School of Engineering)*, 2013.

[45] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*, pp. 52–59, Springer, 2011.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[47] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[49] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[50] Y. Liu, B. Stratman, and S. Mahadevan, "Fatigue crack initiation life prediction of railroad wheels," *International Journal of Fatigue*, vol. 28, no. 7, pp. 747–756, 2006.

[51] N. Dutta, "Geopressure prediction using seismic data: Current status and the road ahead," *Geophysics*, vol. 67, no. 6, pp. 2012–2041, 2002.

[52] I. Maqsood, M. R. Khan, and A. Abraham, "An ensemble of neural networks for weather forecasting," *Neural Computing & Applications*, vol. 13, no. 2, pp. 112–122, 2004.

[53] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar, "Theory-guided data science: A new paradigm for scientific discovery from data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 10, pp. 2318–2331, 2017.

[54] E. Carrera, M. Cinefra, M. Petrolo, and E. Zappino, *Finite Element Analysis of Structures Through Unified Formulation*. Wiley Online Library, 2014.

[55] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International Conference on Machine Learning*, pp. 7354–7363, PMLR, 2019.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[57] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[58] W. D. Pilkey and W. D. Pilkey, *Formulas for stress, strain, and structural matrices*, vol. 107. John Wiley & Sons New Jersey, 2005.

[59] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *ArXiv Preprint ArXiv:1711.05101*, 2017.

[60] S. Huang, Z. Lu, R. Cheng, and C. He, "Fapn: Feature-aligned pyramid network for dense image prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 864–873, 2021.

[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

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

[63] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Inferring solutions of differential equations using noisy multi-fidelity data," *Journal of Computational Physics*, vol. 335, pp. 736–746, 2017.

[64] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Machine learning of linear differential equations using gaussian processes," *Journal of Computational Physics*, vol. 348, pp. 683–693, 2017.

[65] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Numerical gaussian processes for time-dependent and nonlinear partial differential equations," *SIAM Journal on Scientific Computing*, vol. 40, no. 1, pp. A172–A198, 2018.

[66] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[67] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *ArXiv Preprint ArXiv:1711.10561*, 2017.

[68] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[69] M. Vahab, E. Haghighat, M. Khaleghi, and N. Khalili, "A physics-informed neural network approach to solution and identification of biharmonic equations of elasticity," *Journal of Engineering Mechanics*, vol. 148, no. 2, p. 04021154, 2022.

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

[71] D. Chen, Y. Li, K. Liu, and Y. Li, "A physics-informed neural network approach to fatigue life prediction using small quantity of samples," *International Journal of Fatigue*, vol. 166, p. 107270, 2023.

[72] J. Bai, T. Rabczuk, A. Gupta, L. Alzubaidi, and Y. Gu, "A physics-informed neural network technique based on a modified loss function for computational 2d and 3d solid mechanics," *Computational Mechanics*, vol. 71, no. 3, pp. 543–562, 2023.

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

[74] E. Zhang, M. Dao, G. E. Karniadakis, and S. Suresh, "Analyses of internal structures and defects in materials using physics-informed neural networks," *Science Advances*, vol. 8, no. 7, p. eabk0644, 2022.

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

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

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

[78] C. Xu, B. T. Cao, Y. Yuan, and G. Meschke, "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*, vol. 405, p. 115852, 2023.

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

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

[81] S. Das, S. Dutta, C. Putcha, S. Majumdar, and D. Adak, "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*, vol. 6, no. 2, p. 04020013, 2020.

[82] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "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.

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

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

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

[86] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," *ArXiv Preprint ArXiv:2010.08895*, 2020.

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

[88] H. Bolandi, X. Li, T. Salem, V. N. Boddeti, and N. Lajnef, "Deep learning paradigm for prediction of stress distribution in damaged structural components with stress concentrations," *Advances in Engineering Software*, vol. 173, p. 103240, 2022.

[89] M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, "Deep learning of vortex-induced vibrations," *Journal of Fluid Mechanics*, vol. 861, pp. 119–137, 2019.

[90] A. Astaneh-Asl, "Gusset plates in steel bridges–design and evaluation," *Steel TIPS Report, Structural Steel Educational Council Technical Information & Product Services. Moraga, CA*, 2010.

[91] N. R. Ke, S. Chiappa, J. Wang, J. Bornschein, T. Weber, A. Goyal, M. Botvinic, M. Mozer, and D. J. Rezende, "Learning to induce causal structure," *ArXiv Preprint ArXiv:2204.04875*, 2022.