GRAPH NEURAL NETWORKS AS SURROGATE MODELS FOR STRUCTURAL ANALYSIS: A STUDY ON BUCKLING BEHAVIOR

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖMER KURT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

JANUARY 2025

Approval of the thesis:

GRAPH NEURAL NETWORKS AS SURROGATE MODELS FOR STRUCTURAL ANALYSIS: A STUDY ON BUCKLING BEHAVIOR

submitted by **ÖMER KURT** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun Dean, Graduate School of **Natural and Applied Sciences**	_______________
Prof. Dr. Serkan Dağ Head of Department, **Mechanical Engineering**	_______________
Assoc. Prof. Dr. Ulaş Yaman Supervisor, **Mechanical Engineering, METU**	_______________

Examining Committee Members:

Prof. Dr. Melik Dölen Mechanical Engineering, METU	_______________
Assoc. Prof. Dr. Ulaş Yaman Mechanical Engineering, METU	_______________
Prof. Dr. Serkan Dağ Mechanical Engineering, METU	_______________
Assist. Prof. Dr. Orkun Özşahin Mechanical Engineering, METU	_______________
Prof. Dr. Oğuzhan Yılmaz Mechanical Engineering, Gazi University	_______________

Date:10.01.2025

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname:	Ömer Kurt
Signature :	

ABSTRACT

GRAPH NEURAL NETWORKS AS SURROGATE MODELS FOR STRUCTURAL ANALYSIS: A STUDY ON BUCKLING BEHAVIOR

Kurt, Ömer
M.S., Department of Mechanical Engineering
Supervisor: Assoc. Prof. Dr. Ulaş Yaman

January 2025, 127 pages

This thesis presents a novel approach to structural analysis using Graph Neural Networks (GNNs) as surrogate models, specifically focusing on predicting buckling behavior of thin-walled structures with and without stiffeners. The research addresses the computational challenges in traditional finite element analysis by developing an efficient machine learning framework that maintains accuracy while achieving computational speeds faster than conventional methods. To create a well balanced dataset, a comprehensive data generation pipeline is introduced, creating diverse structural geometries using Bezier curves and implementing systematic load case generation procedures. The study developed an enhanced graph representation system that effectively captures both local and global structural behaviors through innovative features such as super-nodes and virtual edges, while ensuring rotational and translational invariance through principal component analysis-based coordinate transformation. The framework demonstrates remarkable accuracy in buckling prediction across both non-stiffened and stiffened structures, with performance validated against finite element analysis results using multiple test datasets, including scaled geometries and complex loading scenarios. The framework reduces analysis time and enables rapid evaluation of multiple design variants. This reduction in computational time, combined with maintained prediction accuracy, demonstrates the potential of the framework to transform preliminary design processes. The research contributes to the growing field of machine learning in structural analysis by providing a robust methodology for creating efficient surrogate models and by demonstrating effectiveness of GNNs on a global property prediction like buckling.

Keywords: Graph Neural Networks, Surrogate Model, Machine Learning, Buckling, Structural Analysis.

ÖZ

YAPISAL ANALİZ İÇİN VEKİL MODELLER OLARAK GRAFİK SİNİR AĞLARI: BURKULMA DAVRANIŞI ÜZERİNE BİR ÇALIŞMA

Kurt, Ömer
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ulaş Yaman

Ocak 2025 , 127 sayfa

Bu tez, Grafik Sinir Ağlarını (GSA) vekil modeller olarak kullanarak yapısal analize yönelik yeni bir yaklaşım sunmaktadır. Çalışma özellikle takviyeli ve takviyesiz ince cidarlı yapıların burkulma davranışlarının tahminini ele almaktadır. Araştırma, geleneksel yöntemlere kıyasla daha hızlı hesaplama süreleri elde ederken doğruluğu koruyan etkili bir makine öğrenimi modeli geliştirerek, geleneksel sonlu elemanlar analizindeki hesaplama zorluklarının çözümünü amaçlamaktadır. Dengeli bir veri seti oluşturmak için, Bezier eğrilerini kullanarak çeşitli yapısal geometriler oluşturan ve sistematik yük durumu üretme prosedürlerini uygulayan kapsamlı bir veri üretim hattı sunulmuştur. Çalışma, süper düğümler ve sanal kenarlar gibi yenilikçi özellikler aracılığıyla hem yerel hem de global yapısal davranışları etkili bir şekilde yakalayan ve temel bileşen analizi tabanlı koordinat dönüşümü yoluyla dönme ve öteleme değişmezliğini sağlayan gelişmiş bir grafik temsil sistemi geliştirmiştir. Geliştirilen model burkulma tahmininde hem takviyesiz hem de takviyeli yapılar için kayda değer bir doğruluk göstermekte olup, performansı ölçeklendirilmiş geometriler ve karmaşık yükleme senaryoları dahil olmak üzere çoklu test veri setleri kullanılarak sonlu elemanlar analizi sonuçlarına göre doğrulanmıştır. Model analiz süresini azaltmakta ve birden fazla tasarım varyantının hızlı değerlendirilmesini sağlamaktadır. Hesaplama süresindeki bu azalma ve korunan tahmin doğruluğu, modelin ön tasarım süreçlerini dönüştürme potansiyelini göstermektedir. Bu araştırma, verimli vekil modeller oluşturmak için etkili bir metodoloji sağlayarak ve Grafik Sinir Ağlarının burkulma gibi global bir özellik tahminindeki etkinliğini göstererek yapısal analizdeki büyüyen makine öğrenimi alanına katkıda bulunmaktadır.

Anahtar Kelimeler: Grafik Sinir Ağları, Vekil Model, Makine Öğrenmesi, Burkulma, Yapısal Analiz.

To my beloved wife and family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Dr. Ulaş Yaman, for his guidance, criticism, support and understanding. His support enabled me to work on this interesting field of research..

I was fortunate to be friend with Yaman Arslanca and Burak Baydar. Their technical and moral support, along with our lengthy yet enlightening conversations, have been invaluable.

I also acknowledge that the numerical calculations reported in this thesis were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

I am deeply grateful to my family for their unconditional love, constant encouragement and continued support. Their unwavering belief in me has been an endless source of strength throughout this endeavor.

Last but certainly not least, I extend my heartfelt thanks to my beloved wife. Words can not express my gratitude for her never-ending support, exceptional patience and cheerful presence in my life. This thesis would not have been possible without her remarkable support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
ANN	Artificial Neural Network
BC	Boundary Condition
BDF	Bulk Data Format (NASTRAN)
CBAR	Bar Element (NASTRAN)
CNN	Convolutional Neural Network
CQUAD4	Four-Node Shell Element (NASTRAN)
CTRIA3	Three-Node Shell Element (NASTRAN)
DOF	Degree of Freedom
EA-GNN	Edge Augmented Graph Neural Network
EIGRL	Real Eigenvalue Extraction Method (NASTRAN)
FEA	Finite Element Analysis
FEM	Finite Element Method
GNN	Graph Neural Network
GP	Grid Point
GPU	Graphics Processing Unit
JIT	Just-In-Time
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MLP	Multi-Layer Perceptron
MPNN	Message Passing Neural Network
MSE	Mean Squared Error
OP2	Output 2 (NASTRAN binary output file format)
PCA	Principal Component Analysis

PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
RBF	Radial Basis Function
RELU	Rectified Linear Unit
RMSE	Root Mean Square Error
RSM	Response Surface Methodology
SAGEConv	GraphSAGE Convolution Layer
SOL	Solution Sequence (NASTRAN)
SPC	Single Point Constraint
TRUBA	Turkish National e-Science e-Infrastructure

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Structural analysis is a critical component in designing efficient and safe engineering structures, particularly in aerospace applications where the balance between structural integrity and weight efficiency directly impacts operational performance and costs. In these applications, thin-walled structures are extensively used due to their excellent strength-to-weight ratios. However, these structures are susceptible to buckling failure - a phenomenon where a structure suddenly loses its stability under compressive loads, often at stress levels well below the material's yield strength [1].

The significance of buckling analysis in aerospace design cannot be overstated. Studies have shown that buckling-related failure modes account for approximately 50% of an aircraft's structural weight as shown in Table 1.1 [2]. This dominance of buckling in structural design creates a pressing need for accurate and efficient analysis methods. However, current industry-standard approaches, primarily based on hand calculations and finite element analysis (FEA) [3, 4], present significant computational challenges. A typical aerospace component might require analysis under thousands of different loading conditions, with each analysis taking considerable computational time [5]. This computational burden becomes particularly acute during preliminary design phases, where rapid evaluation of multiple design variants is crucial for optimization.

The challenge is further complicated when structures incorporate stiffening elements - a common practice in aerospace design to enhance buckling resistance [6]. While stiffeners effectively improve structural performance, they significantly increase the

Failure Mode	% Structural Weight	
	Airplane 1	Airplane 2
Tensile Strength	30.1	18.6
Compressive Strength	0.0	3.5
Crippling[1]	14.3	19.5
Compression Surface Column Buckling[1]	8.1	9.7
Shear or Compression Buckling[1]	19.7	18.1
Aeroelastic Stiffness	14.1	11.6
Durability & Damage Tolerance	13.7	19.0
Total:	100.0	100.0

Table 1.1: Aircraft Structural Weight Breakdown by Failure Mode with [1] Denoting Buckling Failure Modes

complexity of analysis, adding to the computational overhead. Traditional FEA approaches, while accurate, can require hours or even days to analyze complex stiffened structures under multiple load cases, creating a bottleneck in the design process.

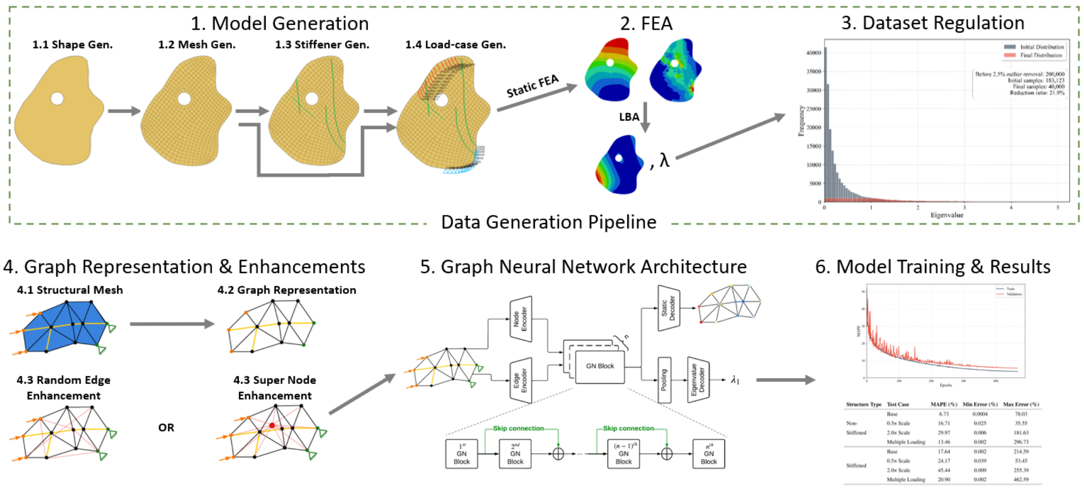
This computational burden presents a clear problem: how can we maintain the accuracy necessary for critical aerospace applications while dramatically reducing analysis time? The need for a solution becomes particularly urgent when considering modern design practices that emphasize rapid iteration and optimization. Engineers need tools that can provide quick, reliable predictions of structural behavior, especially buckling performance, without sacrificing the accuracy that safety-critical applications demand.

This challenge motivates our research into developing efficient surrogate models for structural analysis. The potential impact of such models is significant - reducing analysis time from hours to minutes could transform the design process, enabling more comprehensive exploration of design spaces and ultimately leading to more optimized structures. The development of such tools would not only accelerate the design process but could also enable new approaches to structural optimization that are currently impractical due to computational constraints.

Researchers have explored various surrogate modeling techniques that could potentially replace or complement traditional FEA [7]. Popular machine learning architectures like Convolutional Neural Networks or Recurrent Neural Networks are inherently unsuitable for these tasks due to the complex nature of mesh-based structural simulation. The irregular connectivity and varying node counts in finite element meshes make it impossible to directly apply these traditional deep learning approaches. While various surrogate modeling approaches have been explored to address these challenges, Graph Neural Networks (GNNs) offer a particularly promising direction due to their natural alignment with mesh-based analysis methods. The finite element mesh used in structural analysis can be directly interpreted as a graph, where nodes represent mesh vertices and edges represent element connectivity. This natural correspondence enables GNNs to learn directly from the mesh structure while preserving important geometric and physical relationships. Unlike traditional machine learning approaches that often struggle with irregular mesh structures and varying geometries, GNNs can handle any meshed geometry, making them especially suitable for developing versatile surrogate models for structural analysis that can generalize across different shapes and configurations[8, 9, 10, 11, 12].

Traditional machine learning approaches in structural analysis have largely focused on parametric studies, where models are trained on simplified parameter sets representing specific geometric configurations. These approaches, while useful for their intended parameter spaces, cannot generalize to arbitrary geometries outside their training domain. The structural engineering field needs more versatile surrogate models that can analyze any shape regardless of its geometry without requiring explicit parametrization. Generating a general purpose surrogate model with GNNs is another motivation of this work.

A notable gap exists in the prediction of global structural properties like buckling behavior using machine learning approaches. While local properties like stress and displacement fields have been extensively studied [11, 12], the prediction of global instability phenomena remains largely unexplored. This research was motivated by the opportunity to demonstrate how GNNs, enhanced with specialized architectures like virtual edges and super nodes, can effectively capture and predict such global behavioral properties. These architectural innovations enable better information flow across the structure, making it possible to predict phenomena that depend on the entire structural response rather than just local interactions. This advancement in global property prediction represents a significant step forward in developing comprehensive surrogate models for structural analysis.



Structure Type	Test Case	MAPE (%)	Min Error (%)	Max Error (%)
Non-Stiffened	Base	6.73	0.0004	78.03
	0.5× Scale	16.71	0.025	35.55
	2.0× Scale	29.97	0.006	181.63
	Multiple Loading	13.46	0.002	296.73
Stiffened	Base	17.64	0.002	214.59
	0.5× Scale	24.17	0.039	53.45
	2.0× Scale	45.44	0.009	255.39
	Multiple Loading	20.90	0.002	462.59

Figure 1.1: Overview of the study

1.2 Aim of the Study

This thesis aims to develop a robust Graph Neural Network-based framework for structural analysis that can effectively predict both local and global structural behaviors, with particular emphasis on buckling analysis. Our primary goal is to create a surrogate model that combines the speed advantages of machine learning with the accuracy requirements of engineering applications, focusing specifically on addressing the challenges of time-independent structural analysis problems.

The first fundamental objective is the development of a comprehensive data generation and preparation pipeline. This involves:

- Creating a sophisticated shape generation system using Bezier curves that produces diverse yet physically meaningful structural geometries
- Implementing systematic load case generation procedures that ensure comprehensive coverage of realistic loading scenarios

- Developing automated mesh generation and validation processes that maintain high-quality finite element discretizations
- Establishing rigorous data filtering and balancing methodologies to ensure effective model training

The second major objective is to advance the state-of-the-art in GNN architectures for structural analysis through several key innovations:

- Development of a novel coordinate transformation approach based on principal component analysis to ensure rotational and translational invariance
- Implementation of enhanced information flow mechanisms through virtual edges and super nodes to capture both local and global structural behaviors
- Design of specialized pooling strategies optimized for different prediction tasks, particularly focusing on global properties like buckling eigenvalues
- Creation of robust feature engineering approaches for both node and edge attributes that effectively capture structural characteristics

A crucial research objective is to demonstrate the framework's effectiveness in predicting buckling behavior, a complex global structural property that requires understanding the entire structure's response. This includes:

- Achieving accurate predictions of critical buckling loads for both non-stiffened and stiffened structures
- Demonstrating the model's ability to generalize across different geometric scales and configurations
- Validating the framework's performance against traditional finite element analysis results
- Establishing the computational advantages of our approach compared to conventional methods

Additionally, this research aims to contribute to the broader field of scientific machine learning by:

- Providing insights into effective graph representations for physics-based problems
- Developing methodologies for handling both local and global physical behaviors in graph neural networks
- Creating publicly available datasets and implementation guidelines to facilitate further research
- Establishing best practices for balancing computational efficiency with prediction accuracy in surrogate modeling

Through these objectives, we seek to demonstrate that Graph Neural Networks can serve as reliable and efficient surrogate models for structural analysis, capable of handling complex geometries and loading conditions while maintaining engineering-grade accuracy. The successful achievement of these aims would represent a significant advancement in the application of machine learning to structural engineering problems, potentially transforming how preliminary design and optimization tasks are performed and providing engineers with a tool that maintains high accuracy while significantly reducing computational time.

1.3 Scope of the Study

The scope of this study is carefully defined to enable a focused investigation of GNN-based surrogate models for structural analysis, with particular emphasis on buckling prediction. This scope is outlined across five key aspects.

1.3.1 Geometric Scope

This study focuses on 2D thin-walled structures that can be effectively represented using shell elements. The geometric complexity ranges from simple shapes to more complex configurations containing cutouts and stiffeners. To maintain consistency in the training dataset and ensure meaningful comparisons, all structures are constrained within specific size ranges from 700mm to 1000mm. The scope explicitly excludes 3D structures and complex assemblies to maintain tractability of the problem and focus on fundamental structural behaviors.

1.3.2 Analysis Types

Two primary types of structural analysis are considered in this study. The first is linear static analysis, which focuses on predicting displacement fields. The second is linear buckling analysis, which aims to predict critical buckling load (eigenvalue) of the first buckling mode. The study explicitly excludes more complex analysis types such as non-linear analysis, dynamic analysis, higher-order buckling modes, and post-buckling behavior, as these would significantly increase the complexity of the problem and computational requirements.

1.3.3 Material and Property Considerations

The material scope is limited to isotropic, linear elastic materials, specifically focusing on general purpose aluminum alloy with standard properties (Young's modulus: 76 GPa, Poisson's ratio: 0.3 [13]). Shell elements are modeled with constant thickness throughout the structure. This simplified material model excludes more complex considerations such as composite materials, non-linear material behavior, temperature-dependent properties, and variable thickness distributions, allowing the research to focus on fundamental structural behavior patterns.

1.3.4 Loading and Boundary Conditions

The study encompasses a range of loading conditions including in-plane compression and tension loads, shear loads, and various combinations thereof. Boundary conditions are limited to fixed constraints applied at specified locations. A systematic load case generation procedure is implemented within defined parameter ranges to ensure comprehensive coverage of realistic loading scenarios. The scope excludes out-of-plane loads, thermal loads, dynamic loads, pressure loads, and complex boundary conditions that would require contact analysis, maintaining focus on primary structural responses.

This clearly defined scope allows for a thorough investigation of the research objectives while maintaining feasibility and focusing on the most relevant aspects of the problem. The boundaries established here ensure that the research remains focused and manageable while still providing meaningful contributions to the field of structural analysis and machine learning.

1.4 Contributions and Novelties

This thesis makes several significant contributions to the fields of structural analysis and machine learning, particularly in developing Graph Neural Networks as surrogate models for structural analysis. The key contributions can be summarized in three main areas.

1.4.1 Data Generation and Processing Framework

The first major contribution is the development of a comprehensive data generation and processing pipeline for structural analysis problems. We introduce a novel shape generation system using Bezier curves that creates diverse yet physically meaningful structural geometries, including features like cutouts and stiffeners. A key innovation is our approach to stiffener layout generation and representation, which enables the study of stiffener effects on structural behavior, particularly buckling. The pipeline includes sophisticated load case generation and filtering mechanisms that ensure the dataset captures a wide range of structural behaviors while maintaining physical relevance.

Furthermore, our commitment to making these datasets publicly available, along with comprehensive documentation of generation parameters and preprocessing procedures, represents a significant contribution to the research community. This will enable other researchers to develop and validate their approaches using high-quality, well-characterized data.

1.4.2 Enhanced Graph Representation System

The second major contribution lies in our novel approach to representing structural analysis problems as graphs. We develop multiple strategies for virtual edge creation, including a super node concept that enhances the model's ability to capture global structural behavior. A significant innovation is our method for representing 1D stiffener elements within the graph structure, enabling the model to learn complex interactions between stiffeners and the base structure. We implement a PCA-based coordinate transformation system that ensures rotational and translational invariance, crucial for structural analysis applications. This representation system effectively bridges the gap between traditional finite element analysis and modern graph neural networks.

Our investigation of different information flow mechanisms, particularly the comparison between random virtual edge and super node approaches, provides valuable insights into the architectural requirements for capturing global structural behavior. Through extensive experimentation and analysis, we demonstrate the superior performance of the super node approach for buckling eigenvalue prediction, establishing design principles for future graph-based structural analysis models.

1.4.3 Applications and Practical Impact

The third major contribution is the demonstration of practical applications and potential impact of our approach. Most significantly, we show how our GNN-based surrogate model can revolutionize the stiffener layout optimization process, a traditionally challenging problem with high computational costs. Our model enables rapid evaluation of different stiffener configurations, providing a pathway to explore vast design spaces efficiently. The framework's ability to simultaneously predict both static and buckling behavior, with validated accuracy against FEA results, makes it a valuable tool for preliminary design and optimization studies. The significantly reduced computational time compared to traditional FEA, while maintaining acceptable accuracy, enables new possibilities in structural optimization and design exploration.

These contributions not only advance the theoretical understanding of GNN applications in structural analysis but also provide practical tools for engineering design workflows. The framework developed in this thesis lays the groundwork for future research in structural optimization, particularly in the challenging area of stiffener layout design, while offering immediate practical benefits for engineering applications.

1.5 The Outline of the Thesis

This thesis is organized into six chapters that systematically present the development, implementation, and validation of Graph Neural Networks as surrogate models for structural analysis, with a particular focus on buckling prediction. The structure and content of each chapter are as follows:

Chapter 1 introduces the research motivation, problem definition, and objectives. It presents the fundamental challenges in traditional structural analysis approaches, particularly the computational burden of finite element analysis for design iteration and optimization. The chapter establishes the potential of Graph Neural Networks as surrogate models and outlines the scope of the study, focusing on 2D thin-walled structures and linear analysis types. The chapter concludes by presenting the key contributions of the research.

Chapter 2 provides the theoretical foundation necessary for understanding both traditional structural analysis methods and modern machine learning approaches. It begins with a review of the finite element method. The chapter then presents the evolution and fundamental concepts of Graph Neural Networks, including their message passing mechanisms and recent applications in scientific computing. The literature review systematically covers the progression from traditional surrogate modeling approaches to current state-of-the-art GNN applications in structural analysis, identifying research gaps and opportunities.

Chapter 3 details the comprehensive dataset generation and preprocessing methodology. It begins with the sophisticated shape generation system using Bezier curves, explaining the control point generation and geometric validation processes. The chapter then describes the finite element mesh generation procedures and the approach to load case generation. A significant portion focuses on the dataset balancing methodology, explaining how the initial datasets having hunders of thousands cases were refined to create balanced datasets (40,000 and 80,000 cases respectively) with uniform eigenvalue distributions. The chapter concludes with detailed discussions of the data preprocessing steps, including coordinate frame normalization and feature scaling.

Chapter 4 presents the development of the graph representation system and the neural network architecture. It begins by detailing the feature engineering process for both nodes and edges, including the novel approach to representing stiffener elements. The chapter then explores two key mechanisms for enhancing information flow: random virtual edges and the super node architecture. The neural network architecture section presents both the GraphSAGE and CustomGNN implementations, detailing the message passing mechanisms, pooling strategies, and decoder architectures. Special attention is given to the design decisions that enable effective prediction of both local and global structural properties.

Chapter 5 presents the experimental results and analysis of the developed framework. It begins with the training methodology, including hyperparameter optimization results for the non-stiffened case and their application to stiffened structures. The chapter provides comprehensive performance analysis for buckling prediction, comparing different architectural choices (particularly super node versus random virtual edges) and analyzing their effectiveness. The results section includes detailed analysis of model performance on various test cases, including scaled geometries and multiple boundary condition scenarios. The chapter also discusses the challenges encountered in static displacement prediction.

Chapter 6 concludes the thesis by synthesizing the key findings and contributions. It discusses the implications of the research for practical engineering applications, particularly in the context of preliminary design and optimization of stiffened structures.

The chapter reflects on both the achievements and limitations of the current approach, providing insights for future research directions in the application of geometric deep learning to structural analysis problems.

This structure ensures a logical progression from theoretical foundations through implementation to results and validation, while maintaining focus on the primary research objective of developing effective surrogate models for structural analysis

CHAPTER 2

BACKGROUND

The development of surrogate models for structural analysis using Graph Neural Networks requires a solid understanding of both traditional engineering methods and modern machine learning approaches. This chapter provides the theoretical foundation necessary to comprehend the intersection of these fields and their application in this research. We begin by examining the fundamental principles of finite element analysis, particularly focusing on linear static and linear buckling analysis, which form the basis of our target predictions. Following this, we explore Graph Neural Networks, including their architecture, message passing mechanisms and their suitability for processing mesh-based structural data. The chapter concludes with a comprehensive review of relevant literature, examining both traditional approaches to structural analysis and recent advances in machine learning applications within this domain.

The integration of these fields - structural mechanics, finite element analysis, and geometric deep learning - represents a novel approach to structural analysis. Understanding each component's theoretical basis is crucial for appreciating how they can be effectively combined to create efficient and accurate surrogate models. This chapter aims to provide this foundational knowledge while establishing the context for the methodological innovations presented in subsequent chapters.

2.1 Finite Element Method

The Finite Element Method (FEM) is a well-established numerical technique for solving complex engineering problems. In structural mechanics, it involves discretizing a continuous domain into a finite number of elements, solving governing equations for each element, and assembling these solutions to obtain the global response. For this research, we focus on two specific types of finite element analysis: linear static analysis and linear buckling analysis, as they form the basis of our prediction targets for the Graph Neural Network models.

2.1.1 Linear Static Analysis

Linear static analysis assumes a linear relationship between applied forces and resulting displacements, with small deformation theory and linear elastic material behavior. The fundamental equation governing this analysis is:

$$[K]\{u\} = \{F\} \tag{2.1}$$

where $[K]$ is the global stiffness matrix, $\{u\}$ is the displacement vector, and $\{F\}$ is the applied force vector. In our implementation, we utilize this analysis to acquire nodal displacements and stresses under various loading conditions. The stress tensor components are computed from the displacement field using strain-displacement and stress-strain relationships.

2.1.2 Linear Buckling Analysis

Linear buckling analysis, also known as eigenvalue buckling analysis, predicts the critical loads at which a structure becomes unstable. The analysis basically solves the eigenvalue problem:

$$([K] + \lambda[K_G])\{\phi\} = \{0\} \tag{2.2}$$

where $[K_G]$ is the geometric stiffness matrix, λ is the eigenvalue representing the buckling load factor, and $\{\phi\}$ is the eigenvector representing the buckling mode shape. This analysis is particularly important for thin-walled structures where stability often governs the design.

$$\lambda = \frac{\sigma_{cr}}{\sigma_{applied}} \tag{2.3}$$

Linear buckling analysis with FEM is only possible after linear static analysis is conducted because geometric stiffness matrix is computed using displacement output of linear static analysis. So a linear static analysis prequels linear buckling analysis because of the geometric stiffness matrix dependence to the output of linear static analysis.

Our research focuses on predicting the first (lowest) eigenvalue, as it represents the critical buckling load factor that typically determines the structure's stability limit. This choice aligns with common engineering practice where the primary buckling mode usually drives the design decisions.

For detailed theoretical foundations of the finite element method and its applications in structural analysis, readers are referred to the pivotal works of Zienkiewicz and Taylor [4], Bathe [3], and Cook et al. [14].

2.2 Graph Neural Networks

The emergence of Graph Neural Networks (GNNs) represents a fundamental paradigm shift in computational learning, rooted in the mathematical theory of graph representations and neural information processing. The conceptual foundations can be traced back to early graph theory and network science, with pivotal work by Scarselli et al. [15] introducing the first formal graph neural network model in 2008.

Mathematically, a graph neural network can be formally defined as a function $f_\theta : \mathcal{G} \to \mathcal{Y}$, where $\mathcal{G}$ represents the input graph with node features $\mathbf{X} \in \mathbb{R}^{N \times d}$ and adjacency matrix $\mathbf{A}$, and $\mathcal{Y}$ represents the output space. The core challenge lies in developing a neural network architecture capable of processing graph-structured data while preserving critical topological properties.

2.2.1 Fundamentals of Graph Representation

Graph representations provide a natural framework for describing relationships and interactions between entities in a system. Formally, a graph $G = (V, E)$ consists of a set of vertices (nodes) V and a set of edges $E \subseteq V \times V$ that connect pairs of vertices. An example graph is given in Figure 2.1, where dark blue dots are vertices and light blue lines are edges. In the context of structural analysis, these vertices often represent discretization points or finite elements, while edges capture their physical connectivity and interactions [16].

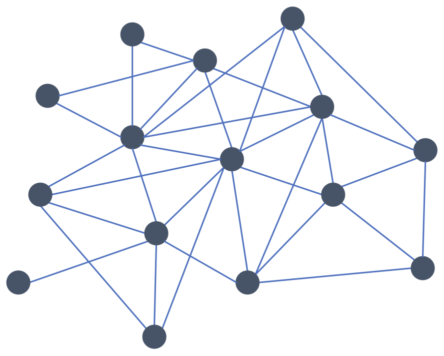


Figure 2.1: An example of graph representation [17]

Each vertex $v_i \in V$ can be associated with a feature vector $\mathbf{h}_i \in \mathbb{R}^d$, where d is the dimension of the feature space. These features encode relevant properties of the vertex, such as spatial coordinates, boundary conditions, or local physical quantities. Similarly, each edge $(i, j) \in E$ connecting vertices i and j can carry its own feature vector $\mathbf{e}_{ij} \in \mathbb{R}^k$, representing properties of the connection such as geometric relationships or physical interactions between the vertices.

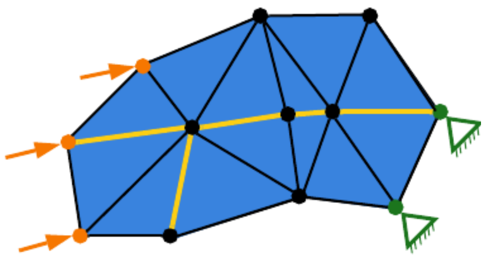
The connectivity structure of the graph is typically represented through an adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$, where $A_{ij} = 1$ if vertices i and j are connected by an edge and $A_{ij} = 0$ otherwise. For weighted graphs, the entries of A can take values other than binary, representing edge weights or strength of connections. The adjacency matrix plays a crucial role in defining how information propagates through the graph during neural message passing operations [8].

In many practical applications, graphs exhibit additional structural properties that can be exploited for more effective learning. For instance, in mesh-based representations, the graph inherits the geometric structure of the underlying physical system. This can be captured through relative positional encodings in edge features, as demonstrated by Pfaff et al. [9], who showed that such geometric information is crucial for learning physical dynamics on meshes.

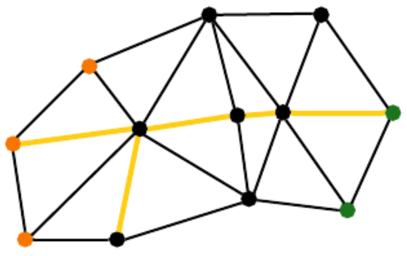
The degree matrix $D = diag(d_1, ..., d_{|V|})$, where $d_i = \sum_j A_{ij}$ represents the degree of vertex i, is often used in conjunction with the adjacency matrix to define the graph Laplacian $L = D - A$. The Laplacian matrix has important spectral properties that relate to the graph's connectivity and can be used to define convolution operations in the spectral domain [18].

An important consideration in graph representation is the handling of different types of relationships between vertices. In structural analysis, this might involve distinguishing between mesh connectivity edges and additional virtual edges introduced to capture long-range interactions. This can be achieved through edge typing or through the use of multiple adjacency matrices, each representing a different class of relationships [9].

The flexibility of graph representations allows for adaptive refinement of the underlying structure. In the context of structural analysis, this might involve adding or removing edges based on physical criteria, or introducing special nodes (such as super-nodes) that can aggregate global information. Such adaptivity is particularly valuable when dealing with complex geometries or when different levels of detail are required in different regions of the structure.



(a) Structural Mesh



(b) Graph

Figure 2.2: Graph representation of a structural mesh

In context of structural engineering, graphs offers natural representation of the mesh data. Figure 2.2 illustrates meshes can be converted directly to graphs, vertexes represent nodes of the shell and edges represent shell element edges.

2.2.2 Evolution of Graph Neural Networks

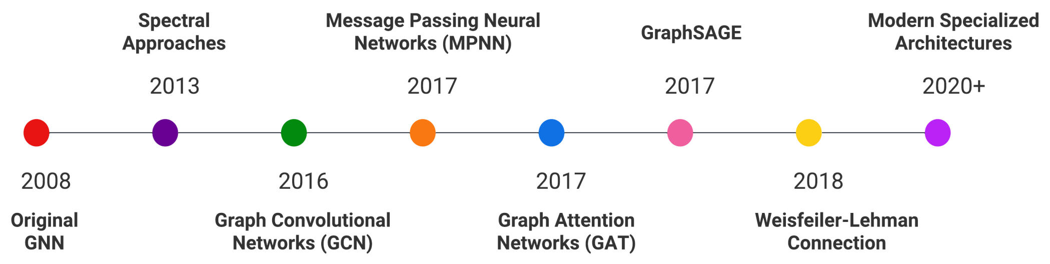

Spectral Approaches
2013
Message Passing Neural Networks (MPNN)
2017
GraphSAGE
2017
Modern Specialized Architectures
2020+
2008
Original GNN
2016
Graph Convolutional Networks (GCN)
2017
Graph Attention Networks (GAT)
2018
Weisfeiler-Lehman Connection

Figure 2.3: Evolution of graph neural network architectures

The development of Graph Neural Networks represents a significant advancement in extending deep learning capabilities to irregular data structures. The timeline is illustrated in Figure 2.3. The foundational work by Scarselli et al. [15] introduced the first Graph Neural Network model, which operated through a recursive neighborhood aggregation scheme until reaching a stable fixed point. While groundbreaking, this early approach faced practical limitations due to its iterative nature and computational demands in achieving convergence. In addition, the inherent complexity of graph-structured data presents significant computational challenges. Unlike regular grid-like data (images, sequences), graphs can have variable connectivity, making traditional deep learning approaches ineffective.

A pivotal advancement came with the introduction of Graph Convolutional Networks (GCNs) by Kipf and Welling [19], who reformulated graph convolutions using a first-order approximation of localized spectral filters. This simplification made training more efficient while maintaining model expressiveness, marking a crucial step toward practical applications. Their work built upon earlier spectral approaches introduced by Bruna et al. [18], who had established the theoretical foundations for extending convolution operations to graph-structured data.

The field further evolved with the development of spatial-domain approaches, which directly operate on graph neighborhoods rather than in the spectral domain. A significant contribution in this direction came from Hamilton et al. [20] with the introduction of GraphSAGE, which enabled inductive learning on previously unseen nodes through learned aggregation functions. This advancement was particularly important for applications where the graph structure could change or where generalization to new graphs was required.

The introduction of attention mechanisms to graph neural networks by Veličković et al. [21] marked another significant milestone. Graph Attention Networks (GATs) enabled the model to assign different importance to different neighbors during the aggregation process, leading to more flexible and powerful architectures. This development aligned with the broader trend of attention mechanisms in deep learning, adapting their benefits to graph-structured data.

A unifying theoretical framework emerged with Message Passing Neural Networks (MPNNs) [22]. This framework generalized many previous approaches and provided a clear theoretical foundation for understanding how information propagates through graph-structured networks. The MPNN framework has become the standard paradigm for designing new graph neural network architectures, offering a clear conceptual model for how these networks learn representations of graph-structured data. Recent theoretical work [23] has focused on understanding the expressive power of different GNN architectures, establishing connections between graph neural networks and the Weisfeiler-Lehman graph isomorphism test. This work has provided crucial insights into the theoretical limitations of current architectures and has guided the development of more powerful variants.

The evolution of Graph Neural Networks continues to be driven by the needs of various applications, particularly in scientific computing and physical simulation. Modern architectures increasingly incorporate domain-specific inductive biases, such as physical constraints or geometric information, leading to more specialized and effective models for particular problem domains. This trend is particularly evident in recent work on mesh-based physical simulations, where graph neural networks have shown remarkable success in learning complex physical dynamics while respecting underlying physical principles.

2.2.3 Message Passing Framework

The message passing framework has emerged as a powerful paradigm for designing GNNs. The core operation in modern GNNs is mostly this message-passing framework. In this framework, each node iteratively updates its representation by aggregating information from its neighbors. This process can be formalized through the Message Passing Neural Network (MPNN) framework [22], which consists of two main phases:

1. **Message Computation**: Each node receives messages from its neighbors, computed as a function of the source and target node features and their edge features.

2. **Node Update**: Each node updates its representation based on its current features and the aggregated messages from its neighbors.

This process can be formally expressed as:

Message Phase:

$$\mathbf{m}_i^{(t+1)} = \sum_{j \in \mathcal{N}(i)} M(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)}, \mathbf{e}_{ij}) \tag{2.4}$$

Update (Aggregation) Phase:

$$\mathbf{x}_i^{(t+1)} = U(\mathbf{x}_i^{(t)}, \mathbf{m}_i^{(t+1)}) \tag{2.5}$$

where $\mathcal{N}(i)$ represents the neighbors of node i, M is the message function, and U is the update function.Both M and U are typically implemented as neural networks. Edge attributes are denoted as $\mathbf{e}_{ij}$ and computed message at the layer t is denoted as $\mathbf{m}_i^{(t+1)}$. $\mathbf{x}_i^{(t+1)}$ and $\mathbf{x}_i^{(t)}$ are updated node and current node attributes respectively.

Message and aggregation phases construct one layer of the GNN network. An illustration of this message passing and aggregation scheme is given in Figure 2.4.

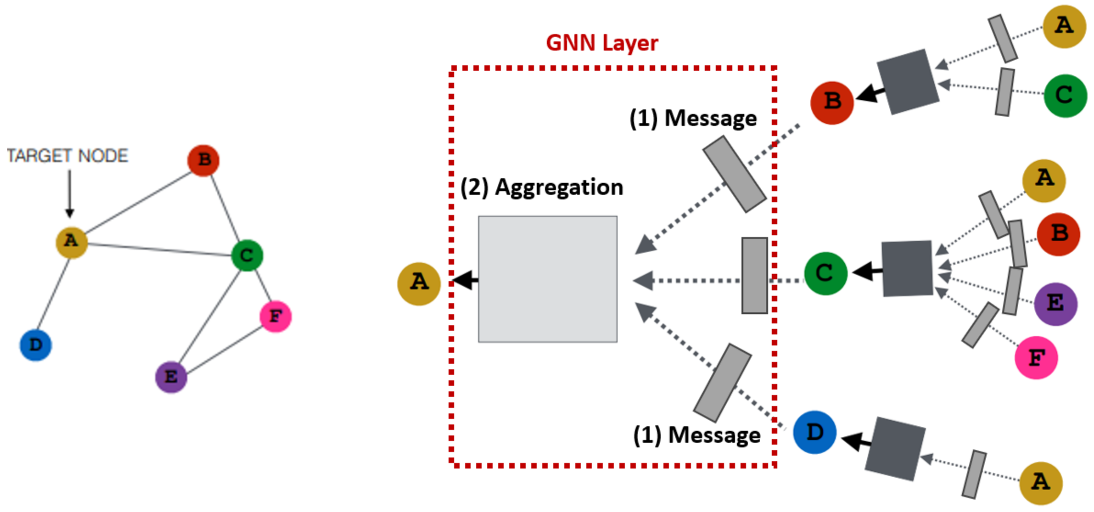


Figure 2.4: Overview of the message passing algorithm. Vertex A was selected and two GNN layers of this vertex were illustrated [16]

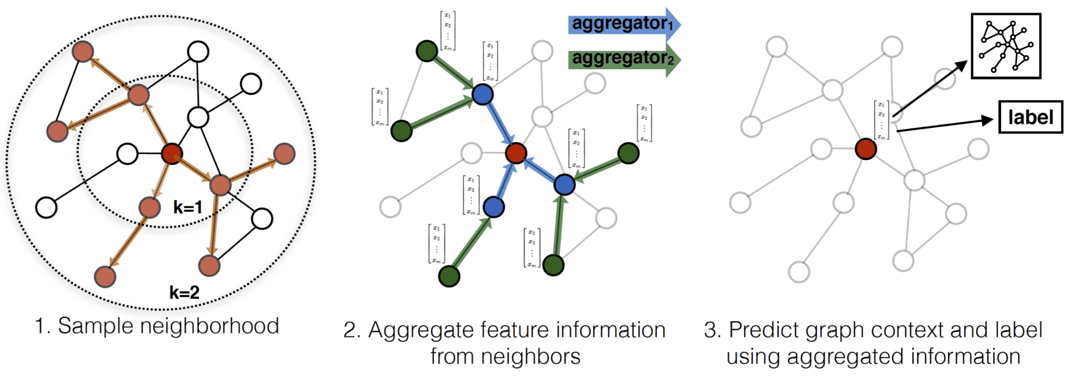
2.2.4 Graph Sampling and Aggregation (GraphSAGE)

In this study, we employ the GraphSAGE architecture [20], which is particularly effective for inductive learning tasks where the model needs to generalize to previously unseen graph structures. GraphSAGE operates by learning how to aggregate neighbor information through a series of learned aggregator functions which illustrated in Figure 2.5. The key operation in GraphSAGE can be expressed as:

$$\mathbf{x}_v^k = \sigma(\mathbf{W}^k \cdot \text{AGGREGATE}_k(\mathbf{x}_u^{k-1}, \forall u \in \mathcal{N}(v))) \tag{2.6}$$

where $\mathbf{x}_v^k$ is the hidden state of node v at layer k, σ is a nonlinear activation function, $\mathbf{W}^k$ is a learnable weight matrix and AGGREGATE_k is an aggregation function that could be mean, sum or max pooling. In our implementation for structural analysis, we utilize the add aggregator as it has shown superior performance in capturing physical relationships.

$$\mathbf{x}_v^k = \sigma(\mathbf{W}^k \cdot (\sum_{u \in \mathcal{N}(v)} \mathbf{x}_u^{k-1})) \tag{2.7}$$



k=1
k=2
1. Sample neighborhood
aggregator$_1$
aggregator$_2$
2. Aggregate feature information from neighbors
label
3. Predict graph context and label using aggregated information

Figure 2.5: Visual illustration of aggregate approach used in GraphSAGE [20]

2.2.5 Applications in Scientific Computing

Graph Neural Networks have emerged as a powerful paradigm in scientific computing, particularly in domains where physical systems exhibit inherent graph-like structures. The ability of GNNs to capture complex relationships while respecting physical constraints has led to significant advances across multiple scientific disciplines. In the field of physics simulation, Sanchez-Gonzalez et al. [8] demonstrated that GNNs can effectively learn to simulate complex physical systems by encoding physical states as nodes and their interactions as edges. Their work showed that GNNs could maintain physical invariances and conservation laws while providing accurate predictions of system dynamics.

A particularly significant advancement in scientific applications came with the development of MeshGraphNets by Pfaff et al. [9]. Their work showed how GNNs could be adapted to operate directly on discretized domains common in engineering and physics simulations. By treating mesh elements as nodes and their connections as edges, they demonstrated that GNNs could learn to predict fluid dynamics, cloth dynamics, and various other physical phenomena with remarkable accuracy. This approach has proven especially valuable in computational fluid dynamics and structural analysis, where mesh-based representations are standard.

In the realm of molecular dynamics and quantum chemistry, the SchNet architecture introduced by Schütt et al. [24] showed how GNNs could model complex quantum interactions at the molecular level. Their approach demonstrated that GNNs could capture both short-range and long-range molecular interactions, leading to accurate predictions of molecular properties and chemical reactions. This success has inspired numerous adaptations in materials science and drug discovery applications.

The effectiveness of GNNs in scientific computing can be attributed to several key properties that align well with physical systems. First, their inherent permutation invariance matches the physical reality that the ordering of elements in a system should not affect its behavior. Second, the message-passing framework naturally captures the locality principle common in many physical systems, where interactions primarily occur between nearby elements but can propagate to create global effects. Finally, GNNs exhibit strong geometric generalization capabilities, allowing them to operate effectively across different scales and discretizations - a crucial feature for scientific applications where problem domains may vary significantly in size and resolution.

Recent work [25] has also shown how GNNs can be enhanced with physics-informed constraints, leading to solutions that better respect underlying physical principles. These physics-informed GNNs combine the flexibility of neural networks with the rigorous constraints of physical laws, resulting in more reliable and interpretable predictions for scientific applications. This synthesis of data-driven learning with physical principles represents a promising direction for future development in scientific computing.

2.2.6 Key Advantages for Scientific Applications

Graph Neural Networks possess several fundamental properties that make them particularly advantageous for scientific applications [16]. The first key advantage lies in their inherent permutation invariance, a property that aligns naturally with physical systems . This invariance ensures that the predictions remain consistent regardless of how the nodes in the graph are ordered or numbered, reflecting the fundamental nature of physical systems where the ordering of elements should not affect the underlying physics. A second crucial advantage is found in the networks' ability to capture both local and compositional behavior through their message passing framework. This architectural feature mirrors the hierarchical nature of many physical phenomena, where complex global behaviors emerge from local interactions. In structural analysis, for instance, local deformations and stress distributions combine to produce overall structural responses. The message passing mechanism allows the network to learn these multi-scale relationships, propagating information from local neighborhoods to capture increasingly larger-scale phenomena with each layer of the network.

The geometric generalization capabilities of GNNs represent another significant advantage for scientific applications. Unlike traditional neural network architectures that often struggle with varying input sizes or geometries, GNNs can naturally handle different mesh resolutions and geometric configurations. This flexibility is particularly valuable in engineering applications where analyses must be performed on structures of varying sizes and complexities. The ability to generalize across different geometric scales and configurations means that a single trained model can be applied to a range of problem sizes without requiring retraining.

Furthermore, GNNs offer the ability to incorporate physical symmetries and conservation laws directly into their architecture. This can be achieved through appropriate choice of message passing functions and update rules, ensuring that the network's predictions respect fundamental physical principles. The ability to embed such physical constraints into the network architecture helps to ensure that predictions remain physically meaningful, even when the network encounters scenarios outside its training distribution.

Finally, the interpretability of GNN architectures provides an additional advantage in scientific applications. The message passing operations and node updates can often be related to physical processes, making it easier to understand and validate the network's behavior from a scientific perspective. This interpretability is crucial in scientific and engineering applications where understanding the reasoning behind predictions is often as important as the predictions themselves.

These advantages collectively make GNNs a powerful tool for scientific computing applications, particularly in domains where physical systems can be naturally represented as graphs and where the preservation of physical principles is crucial for meaningful results.

2.3 Literature Review

The intersection of structural analysis and machine learning, particularly in the context of Graph Neural Networks, represents a rapidly evolving field that draws from multiple disciplines. This literature review examines the progression from traditional analytical methods for stiffened structures to modern machine learning approaches, with a particular focus on surrogate modeling techniques. We begin by exploring the fundamental research in stiffened metallic structures and their structural analysis methods, then examine the evolution of surrogate modeling techniques in structural engineering. Finally, we review recent developments in GNN-based approaches for structural analysis, highlighting current capabilities and limitations. This comprehensive review establishes the context for our research and identifies the gaps in current knowledge that our work aims to address.

2.3.1 Stiffened Metallic Structures and Their Structural Analyses

Stiffened metallic structures are composite structural elements consisting of a thin metal plate or shell reinforced by an array of stiffeners. These stiffeners, which can be oriented in one or multiple directions, are typically welded or riveted to the base plate and serve to enhance the structure's load-carrying capacity and stability without significantly increasing its weight. This design approach is particularly prevalent in weight-critical applications such as aircraft fuselages, ship hulls, and bridge decks, where the need to minimize material usage while maintaining structural integrity is paramount [26]. The base plate in stiffened structures primarily resists in-plane and lateral loads through membrane action, while the stiffeners increase the structure's overall bending stiffness and resistance to buckling. Stiffeners can take various cross-sectional forms, with T-sections, L-sections, and Z-sections being the most common configurations in practice [27, 6]. The choice of stiffener geometry significantly influences both the structure's performance and its manufacturing complexity.

Traditional hand calculation methods for analyzing stiffened panels typically employ simplified approaches based on beam-column theory and plate theory. For buckling analysis, these methods often treat the structure as a combination of plate elements between stiffeners and the stiffeners themselves as beam-columns. The classical approach, developed by von Karman and later refined by Timoshenko [1], involves Equation (2.8) and (2.9).

Calculation of the critical buckling stress for the plate between stiffeners:

$$\sigma_{cr} = k\frac{\eta\pi^2 E}{12(1-\nu^2)}\left(\frac{t}{b}\right)^2 \tag{2.8}$$

where k is the buckling coefficient depending on boundary conditions, E is Young's modulus, ν is Poisson's ratio, t is plate thickness, and b is the width between stiffeners. Analysis of stiffener buckling considering both local and global modes:

$$P_{cr} = \frac{\pi^2 EI}{(KL)^2} \tag{2.9}$$

where I is the moment of inertia of the stiffener (including effective width of the plate), L is the stiffener length, and K is the effective length factor.

The development of finite element methods revolutionized the analysis of stiffened structures by enabling more accurate representation of complex geometries and loading conditions. Modern FEM approaches typically employ shell elements for the plate and either shell or beam elements for the stiffeners [4]. The critical aspects of FEM modeling for stiffened structures include:

- Element Selection and Mesh Refinement: The plate is typically modeled using quadrilateral shell elements with both membrane and bending capabilities. Mesh density must be sufficient to capture local buckling modes, typically requiring 4-8 elements between stiffeners. NASTRAN's user guide [28] and a work in the area [29] recommends a minimum of five grid points per half sine wave (buckled shape).

- Stiffener Modelling: 2D shell elements can be used for detailed analysis of local behavior and 1D beam elements can be used for cases where local stiffener deformation is less critical.

- Boundary Condition Modeling: Appropriate representation of support conditions at panel edges, which can be tricky and difficult to estimate because it changes with type and stiffness of stiffeners.

- Load Application: Careful consideration of load introduction methods to avoid artificial stress concentrations and proper representation of distributed and concentrated loads needed.

For buckling analysis, the finite element approach typically involves solving the eigenvalue problem:

$$([K] + \lambda[K_G])\phi = 0 \tag{2.10}$$

where $[K]$ is the linear stiffness matrix, $[K_G]$ is the geometric stiffness matrix, λ is the eigenvalue (buckling load factor), and ϕ is the eigenvector representing the buckling mode shape.

The development of computational methods has also enabled more sophisticated analysis approaches, such as the Equivalent Single Layer (ESL) method proposed by Avi et al. [30]. This method transforms the stiffened panel into an equivalent homogeneous plate with appropriately modified stiffness properties, significantly reducing computational requirements while maintaining acceptable accuracy for global response predictions.

2.3.2 Verification of Linear Buckling Analysis for Thin Walled Slender Structures

While nonlinear analysis methods can provide more accurate results for buckling behavior, especially post-buckling, linear buckling analysis using finite element methods (FEM) has been shown to be suitable for predicting initial buckling loads of thin-walled slender structures under certain conditions. Several studies have verified the applicability of linear buckling FEM for very slender plates and panels through comparisons with experimental results.

Muameleci [29] conducted an extensive study comparing linear buckling FEM results to experimental data for thin steel plates under various loading conditions. For plates with slenderness ratios (width to thickness) greater than 100, he found that linear buckling analysis predicted initial buckling loads within 5% of experimental values. This correlates with other studies [31, 32] on the same topic. Specifically, for 1m x 1m steel plates with thicknesses of 1.5mm and 2mm, the linear FEM results matched experimental buckling loads to within 3%. Similarly, Sun et al. [33] demonstrated even higher accuracy for hat-stiffened composite panels, achieving FEA predictions within 0.5% of experimental strain gauge measurements, with excellent correlation between predicted and observed buckling modes. These studies collectively demonstrate that for very slender plates and panels, with thickness-to-width ratios less than about 1/80, linear buckling analysis can provide reasonably accurate predictions of initial buckling loads and modes. The success of linear methods for such structures is attributed to buckling occurring well within the linear elastic range of the material, before any significant geometric nonlinearity develops. In summary, linear buckling analysis is accurate for slender plates where buckling occurs clearly before the yield limit of the material.

However, it should be noted that linear analysis cannot capture post-buckling behavior or ultimate failure loads. For thicker structures or those expected to experience significant post-buckling strength, nonlinear analysis methods are still recommended. Additionally, imperfections and residual stresses may cause real structures to buckle at lower loads than predicted by linear analysis of perfect geometries.

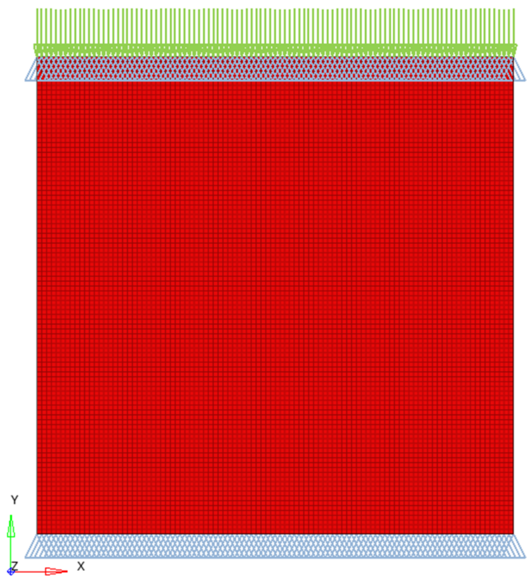


Figure 2.6: Verification FEA model with both end simply supported and 100N force applied. Constrained DoFs of loaded edge are 1 and 3 whereas other edge's contrained DoFs are 1,2 and 3

To verify linear buckling analysis with the theory, the panel given in Figure 2.6 with 1.5mm thickness and 1m x 1m size was modeled and analyzed. Both ends of the panel were simply supported and applied load was 100N. The material was selected as aluminum with 76GPa elastic modulus and 0.3 Poisson ratio [13].

Linear buckling analysis provided an eigenvalue of 2.21. We can convert this value into critical stress using the formula:

$$\sigma_{cr}^{fea} = \frac{\lambda F_{applied}}{A} \tag{2.11}$$

where A is the area ($1.5mm*1000mm = 1500mm^2$), λ is resulted eigenvalue of FEA and σ_{cr}^{fea} is critical buckling stress result using FEA. We calculated $\sigma_{cr}^{fea} = 0.147$. Then we can calculate theoretical critical buckling stress by:

$$\sigma_{cr}^{theory} = k\frac{\eta\pi^2 E}{12(1-\nu^2)}\left(\frac{t}{b}\right)^2 \tag{2.12}$$

where k is 0.93 which is buckling coefficient taken from the reference [34], η is the plasticity factor which is 1 for this problem because part is slender, ν, t and b are given as 0.3, 1.5mm and 1000mm respectively. Using all of these parameters, we calculated the $\sigma_{cr}^{theory} = 0.144$. The error between σ_{cr}^{fea} and σ_{cr}^{theory} is only 2.8%. This verifies the linear buckling analysis result and also shows correlation with the literature.

In summary, for the very slender 1.5mm thick, 1m x 1m panels considered in this thesis, linear buckling FEM is justified based on previous experimental verifications of the method for similar thin-walled structures and theoretical calculations. The linear approach provides a computationally efficient means of predicting initial buckling behavior for parametric studies and optimization, while acknowledging its limitations in capturing full structural response.

2.3.3 Surrogate Models for Structural Analysis

The development of surrogate models, also known as metamodels or reduced-order models, represents a significant advancement in structural analysis, particularly in scenarios requiring numerous evaluations such as optimization or uncertainty quantification. These models aim to approximate the input-output relationship of complex structural analyses while significantly reducing computational costs [5].

2.3.3.1 Traditional Surrogate Modeling Approaches

Traditional surrogate modeling approaches in structural analysis have primarily relied on polynomial response surfaces and interpolation-based methods. Response Surface Methodology (RSM), introduced by Box and Wilson, has been widely applied in structural engineering due to its simplicity and interpretability [35]. However, RSM's effectiveness is limited to problems with relatively few design variables and smooth response surfaces.

More sophisticated interpolation methods emerged to address these limitations. Kriging, also known as Gaussian Process regression, gained popularity in structural analysis due to its ability to provide both predictions and uncertainty estimates [36]. Radial Basis Functions (RBF) offered another powerful approach, particularly effective for highly nonlinear responses [37]. These methods can be expressed mathematically as:

$$\hat{y}_(\mathbf{x}) = \sum_{i=1}^{n} w_i \phi(|\mathbf{x} - \mathbf{x}_i|) \tag{2.13}$$

where $\hat{y}_(\mathbf{x})$ is the predicted response, w_i are the weights, and $\phi(\cdot)$ is the basis function.

2.3.3.2 Emergence of Machine Learning Techniques

The emergence of machine learning techniques has led to more advanced surrogate modeling approaches. Artificial Neural Networks (ANNs) have demonstrated particular success in capturing complex nonlinear relationships in structural analysis. Early applications by Papadrakakis et al. [7] showed that neural networks could effectively approximate structural responses for optimization purposes.

Recent developments have focused on physics-informed machine learning approaches, where physical constraints and domain knowledge are incorporated into the surrogate model architecture. Raissi et al. [38] demonstrated how physics-informed neural networks (PINNs) could solve partial differential equations while respecting underlying physical principles. Their approach minimizes a combined loss function:

$$\mathcal{L} = \mathcal{L}_{data} + \lambda \mathcal{L}_{physics} \tag{2.14}$$

where $\mathcal{L}_{data}$ represents the data-fitting term and $\mathcal{L}_{physics}$ enforces physical constraints.

For stiffened panel analysis specifically, surrogate models have been developed to predict various responses including displacement and stress fields under various loading conditions, critical buckling loads and post-buckling behavior.

Another advancement in surrogate modeling has been the development of multi-fidelity approaches, particularly valuable for stiffened panel analysis. Tian et al. [39] introduced a variable-fidelity surrogate model for predicting shell buckling behavior. Their approach combines high-fidelity finite element results with lower-fidelity analytical solutions, expressed mathematically as:

$$f_h(\mathbf{x}) \approx \rho(\mathbf{x}) f_l(\mathbf{x}) + \delta(\mathbf{x}) \tag{2.15}$$

where f_h and f_l are high and low-fidelity models respectively, $\rho(\mathbf{x})$ is a scaling factor, and $\delta(\mathbf{x})$ is a discrepancy function. This approach has shown particular effectiveness in predicting buckling loads while reducing computational cost by up to 80%.

Limbachiya et al. [40] developed a neural network architecture specifically for predicting shear resistance in cellular steel beams. Their model incorporated both geometric and material nonlinearities, achieving prediction accuracies within 5% of detailed finite element results.

Sun et al. [33] developed two artificial neural network architectures to predict the compression buckling behavior of composite hat-stiffened panels. Their approach used a combination of finite element simulation and experimental verification to generate training data. The neural networks were designed to predict both buckling loads and buckling modes based on four key mechanical parameters: the in-plane tensional stiffness ratio (A11/A22), the orthotropic ratio of the stiffened panel (D1/D2), the tensional stiffness ratio of the stiffener to the skin ((EA)$_s$/(EA)$_p$), and the torsion and bending stiffness ratio (D3/D2). Using an architecture of 4-8-6-2-1 nodes for buckling load prediction and 4-7-2 nodes for buckling mode classification, their models achieved maximum prediction errors of 7.16% for buckling loads and an accuracy of 98.48% for buckling mode prediction on the test dataset. The approach demonstrated robust accuracy compared to traditional engineering calculation methods while maintaining computational efficiency.

Effective feature representations for stiffened panel analysis is of crucial importance and researchers paid particular attention to several key factors that significantly impact model performance. Geometric parameters such as panel aspect ratio, slenderness, stiffener spacing and dimensions, as well as cross-sectional properties are crucial in shaping the behavior of stiffened panels. Additionally, initial imperfection patterns are important for capturing real-world deviations. In terms of loading parameters, factors like load eccentricity, load distribution, and the effects of combined loading play a pivotal role in accurately predicting panel responses. Boundary conditions, including edge restraint coefficients, rotational stiffness parameters, and connection characteristics, further influence the panel's structural performance. These features together form the basis for enhancing the predictive capabilities of stiffened panel models.

2.3.4 State-of-the-Art in GNN-Based Structural Analysis

The application of Graph Neural Networks to structural analysis represents a significant advancement in computational mechanics, particularly addressing the limitations of traditional numerical methods. Early work in this domain focused primarily on simple truss structures, where the graph representation naturally aligned with the physical structure. Nourian et al. [41] demonstrated the effectiveness of GNNs in truss optimization, employing a graph convolutional network approach that directly operated on the structural topology. Their work showed that GNNs could effectively capture both local and global structural behavior, leading to accurate predictions of structural responses while maintaining computational efficiency.

The incorporation of attention mechanisms has further enhanced the capability of GNNs in structural analysis. Veličković et al. [21] demonstrated that attention-based approaches can effectively weight the importance of different structural connections, leading to more accurate predictions particularly in regions of high stress concentration or geometric discontinuity.

The prediction of stress fields and displacement distributions represents a particularly challenging aspect of GNN-based structural analysis. The field prediction capability of GNNs has been demonstrated in various contexts, with Gao et al. [42] introducing a hypergraph message passing approach that showed promising results for fluid dynamics simulations. Their work highlighted the importance of proper feature engineering and message passing mechanisms for field predictions. Also, Barwey et al. [43] proposed a multiscale graph neural network autoencoder that effectively captures field variable distributions across different scales.

Recent developments have shown promising results in various aspects of structural analysis using GNNs. For instance, Chou et al. [10] proposed StructGNN, an efficient framework for static structural analysis that incorporates pseudo nodes as rigid diaphragms in each story. This approach demonstrated remarkable accuracy, achieving over 99% accuracy in predicting displacements, bending moments and shear forces while also showing strong generalization capabilities for taller, previously unseen structures.

A significant advancement in GNN-based structural analysis has been the development of physics-informed approaches that incorporate mechanical principles directly into the neural network architecture. In a recent work [44], it is demonstrated that by integrating fundamental mechanical formulations into the learning process, GNNs can perform accurate structural analysis without requiring labeled data. This approach not only addresses the challenge of limited training data availability but also ensures theoretical correctness of the predictions, a crucial consideration in structural engineering applications. In addition, Shao et al. [25] have focused on incorporating physical constraints and domain knowledge into GNN architectures. They developed a physics-informed GNN framework that explicitly enforced conservation laws and boundary conditions through the network architecture. Their approach combined the flexibility of neural networks with physical constraints, leading to more reliable and interpretable predictions. The mathematical formulation of their physics-informed loss function can be expressed as:

$$\mathcal{L}_{total} = \mathcal{L}_{data} + \lambda_1 \mathcal{L}_{physics} + \lambda_2 \mathcal{L}_{boundary} \tag{2.16}$$

where λ_1 and λ_2 are weighting parameters for the physics and boundary condition constraints respectively.

A breakthrough in applying GNNs to continuous structures came with the development of MeshGraphNets by Pfaff et al. [9]. Their work established a framework for converting meshes into graph representations, enabling the application of GNNs to a broader range of structural analysis problems. The framework introduced the concept of world edges, which are additional graph connections beyond the physical mesh connectivity, to capture long-range interactions within the structure. The original work included only a demonstration of dynamic analysis. But the same architecture was used as base model for structural static analysis in later studies [11]. Pegolotti et al. [45] demonstrated the effectiveness of using virtual edges to connect boundary nodes with internal nodes, enabling more effective propagation of boundary condition information throughout the structure. Their approach showed particular success in problems involving complex mixed boundary conditions.

The development of multi-scale approaches in GNN-based structural analysis has enhanced the capability to handle complex systems. Fortunato et al. [46] introduced Multiscale MeshGraphNets, which effectively capture both local and global structural behavior through a hierarchical graph representation. Their architecture employs a series of graph pooling and unpooling operations, similar to the U-Net architecture commonly used in image processing, but adapted for irregular graph structures. This approach has proven particularly effective for problems where structural responses occur at multiple scales, such as buckling analysis where both local and global modes need to be considered simultaneously.

Recent advances in the field have also focused on improving the interpretability of GNN predictions for structural analysis. Deshpande et al. [47] introduced MAG-NET, a graph U-Net architecture specifically designed for mesh-based simulations. Their work demonstrated that the intermediate representations learned by the network could be interpreted in terms of physical quantities, providing insights into how the network makes its predictions. This interpretability is particularly valuable in engineering applications where understanding the reasoning behind predictions is crucial for safety-critical decisions.

The integration of uncertainty quantification into GNN predictions represents another advancement in the field. Black and Najafi [48] developed a multi-fidelity GNN framework that not only provides predictions of structural responses but also quantifies the uncertainty associated with these predictions. Their approach combines probabilistic graph neural networks with traditional error estimation techniques from finite element analysis, providing a more complete picture of prediction reliability.

The representation of structural systems as graphs has emerged as a critical aspect of GNN implementations. The conventional approach of representing each finite element as a vertex in the graph, while intuitive, can lead to computational inefficiencies as demonstrated by Cai and Jelovica [12]. They introduced an innovative graph embedding approach for 3D stiffened panels that significantly reduces computational requirements while maintaining high prediction accuracy. Their method achieved approximately 96% reduction in training time and 98% reduction in GPU memory usage compared to conventional finite element-vertex embeddings, demonstrating the importance of efficient graph representations in practical applications. Their work showed that for a stiffened panel analysis, the computational complexity of GNN training scales linearly with the number of vertices, making the conventional element-vertex representation computationally expensive for fine meshes. This observation aligns with the findings of Deshpande et al. [47], who noted that the computational burden of message passing increases significantly with mesh refinement.

The ability of GNNs to handle geometric variations and complex boundary conditions has been demonstrated across multiple studies. Research has shown that GNN-based models can effectively predict structural responses under various loading conditions, material behaviors, and geometric configurations [11, 47, 12]. This versatility represents a significant advantage over traditional neural network approaches, which often struggle with geometric variations and require retraining for different structural configurations.

One of the main problem in GNN applications is that long range dependencies cannot be captured in several steps (convolutions) if two vertices are too far apart, many hops away. If convolution number increased to let message propagate to further distances, vanishing gradient problem might happen. Excessive use of GNN layers have this adverse effect on the gradients. To address these computational challenges, several innovative graph representation strategies have emerged. For instance, Lino et al. [49] proposed a multi-scale graph representation that reduces computational complexity while maintaining prediction accuracy. Their approach involves hierarchical graph coarsening, which allows the network to capture both local and global structural behaviors efficiently. Similarly, a recent work [11] introduced an edge-augmented GNN architecture that enhances information flow across the computational domain through strategically placed virtual edges, demonstrating significant improvements in prediction accuracy for time-independent problems. Also residual connections are shown to be reducing the vanishing gradients problem.

Ross and Hambleton [50] introduced an approach to predict mechanical responses of 3D lattice structures using graph neural networks (GNNs). The researchers developed a method to approximate the compressive behavior of lattice geometries by training a graph neural network on a dataset of lattices sampled continuously from the space of all possible lattice configurations. Their key innovation lies in the ability to predict material properties directly from lattice geometry, offering potential for real-time structural feedback during the design stage. By using an encoder-processor-decoder architecture and exploring different data representations, they achieved up to 93% accuracy in predicting lattice compression, demonstrating the potential of machine learning to rapidly assess structural performance across various geometric variations.

Taghizadeh et al. [51] proposed a multifidelity graph neural network (MFGNN) approach for solving partial differential equations (PDEs) in mesh-based simulations. The research addresses the computational challenges of high-resolution structural analyses by developing a hierarchical learning strategy that combines low-fidelity and high-fidelity data. Their framework includes two innovative approaches: a hierarchical multifidelity architecture and a curriculum learning-based training method. Validated across multiple experiments, including 2D stress distribution and complex 3D vehicle aerodynamics simulations, the MFGNN consistently outperformed single-fidelity graph neural networks. The method significantly reduces computational demands while maintaining high accuracy, demonstrating the potential of machine learning to provide efficient and accurate surrogate modeling for complex structural and fluid dynamics problems.

The challenge of ensuring translation and rotation invariance in GNN predictions has been addressed through various approaches. Liu et al. [52] proposed the Invariant Neural Operator (INO) framework that embeds these invariances directly into the network architecture. This approach bears similarities to the coordinate transformation technique employed in [11], where principal component analysis is used to create a normalized coordinate system for all structural analyses.

Recent work has also focused on improving the prediction of global structural responses through enhanced graph connectivity. The super-node concept, as implemented in our research, represents a novel approach to this challenge. This method bears some similarities to the global attention mechanisms proposed by Wu et al. [53], but offers a more computationally efficient solution by creating a single node that connects to all other nodes in the graph. This approach is highly probable to be particularly effective for buckling analysis, where the prediction target is a global property of the structure rather than a local field quantity.

Another advancement in applying GNNs to time-independent problems came with the work of Gladstone et al. [11]. They identified and addressed fundamental challenges in static mechanics problems. They noted that solution at any point depends on points that are farther away in the domain, as solutions can be expressed as integrals over the entire domain. This necessitated efficient message propagation between distant points. In adddition, they observed that conventional mesh-based GNN approaches typically fail at capturing the physics accurately due to the requirement of a large number of message passing steps for information exchange between distant nodes. To address these challenges, they introduced two novel architectures: the Edge Augmented Graph Neural Network (EA-GNN) and the Multi-Graph Neural Network (M-GNN). The EA-GNN introduces "virtual" edges to yield faster information propagation, significantly improving computational efficiency. The M-GNN pursues a multi-resolution approach inspired by the multi-grid method, based on the Graph U-Net architecture. Their work also introduced a novel coordinate transformation technique that enables rotation and translation invariance, facilitating the treatment of variable domains.

2.4 Closure

As aforementioned, current state-of-the-art GNN-Based structural analysis still has several limitations and unexplored areas. While significant progress has been made in static stress analysis and displacement predictions, there remains a notable gap in the literature regarding buckling analysis and the incorporation of 1D stiffener elements in graph-based structural analysis. The prediction of buckling eigenvalues presents unique challenges as it requires the model to capture both local and global structural behavior simultaneously. Furthermore, the effective representation of 1D stiffener elements within the graph structure, while maintaining their physical significance and influence on the overall structural behavior, remains an open challenge.

In conclusion, the application of Graph Neural Networks to structural analysis represents a rapidly evolving field with significant recent advances but also great challenges. The work presented in this thesis, particularly the development of efficient graph representations and the successful prediction of buckling behavior, contributes to this growing body of knowledge. While challenges remain, the demonstrated success in plate buckling analysis and the computational efficiency gains achieved through novel graph representations suggest promising directions for future research.

CHAPTER 3

DATASET GENERATION AND PRE-PROCESSING

3.1 Introduction

This chapter provides information on shape, mesh and load case generation. Details of the shape dataset are given, the procedure for rotation and translation invariance and virtual edge creation is explained and the final setup for pre-processed graph datasets to be used in the experiments is mentioned. Figure 3.1 shows the flowchart of the dataset generation process, which consists of six main stages: shape generation, mesh generation, load case generation, finite element analysis, dataset regularization, and dataset split. Figure 3.2 illustrates the detailed process of dataset regularization and processing, showing how the initial dataset was expanded, balanced, and split into training, validation, and various test sets. The following sections detail each of these stages, their implementation, and their significance in the overall process.

3.2 Shape Generation

The shape generation process is a crucial aspect of this study, as it enables the creation of a diverse and representative dataset to train the graph neural network model. We developed a generative system that produces structural shapes through a sophisticated combination of Bezier curves and control points, allowing for the generation of both simple geometries and more complex shapes with cutouts. The system operates within defined geometric constraints to ensure all generated shapes maintain practical relevance for structural analysis.

The initial dataset generation phase produced 1,000 shapes with cutouts and 3,000

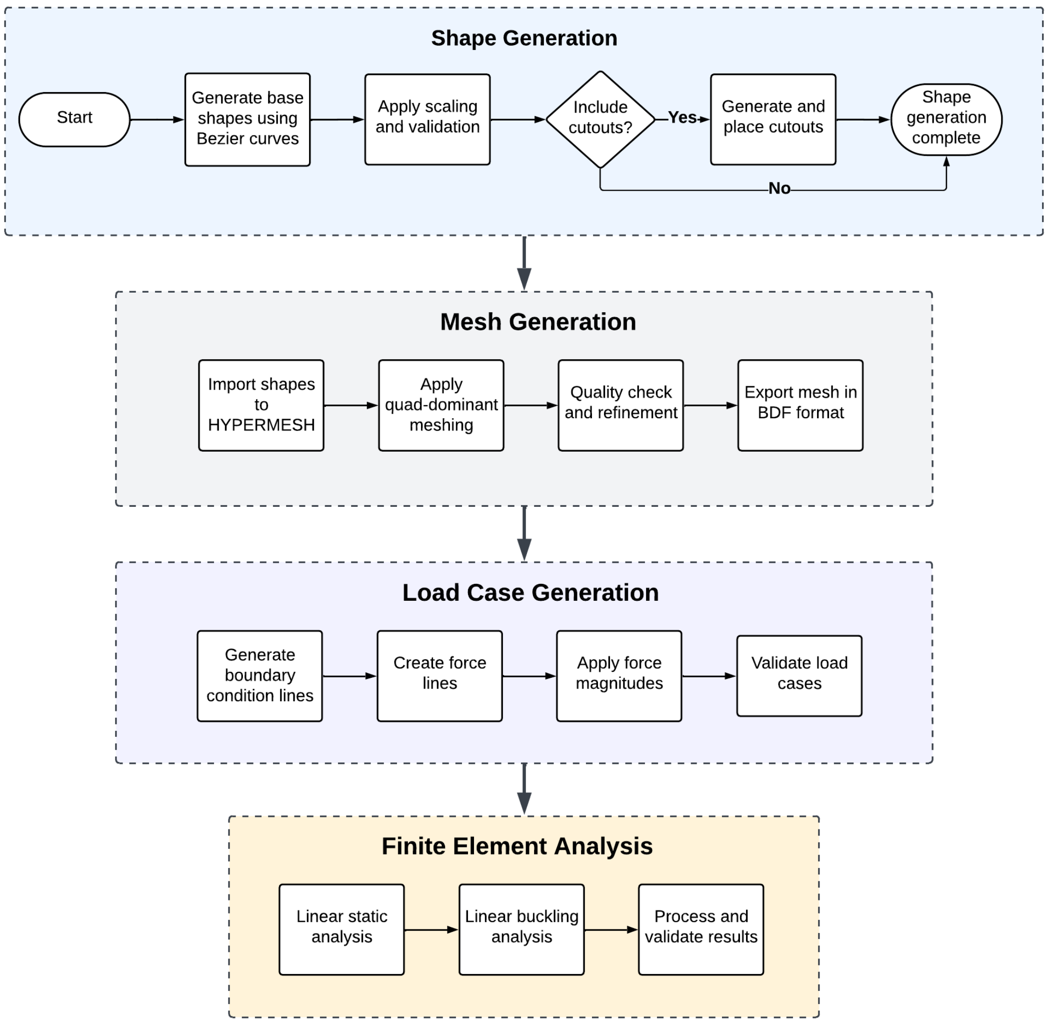

Shape Generation
Start
Generate base shapes using Bezier curves
Apply scaling and validation
Include cutouts?
Yes
Generate and place cutouts
Shape generation complete
No
Mesh Generation
Import shapes to HYPERMESH
Apply quad-dominant meshing
Quality check and refinement
Export mesh in BDF format
Load Case Generation
Generate boundary condition lines
Create force lines
Apply force magnitudes
Validate load cases
Finite Element Analysis
Linear static analysis
Linear buckling analysis
Process and validate results

Figure 3.1: Flowchart for dataset generation

shapes without cutouts. This dataset was used to train static analysis prediction and buckling analysis prediction models. Second dataset generation phase produced 10,000 shapes with cutouts and 10,000 shapes without cutouts. This dataset was used to generate diverse examples for trainings with stiffeners. Shape generation methodology presented in upcoming sections can be summarized as illustrated in Figure 3.3

3.2.1 Base Shape Generation

The shape generation algorithm employs a systematic approach to create organic-asymmetric shapes using Bezier curves. The process begins with the generation of

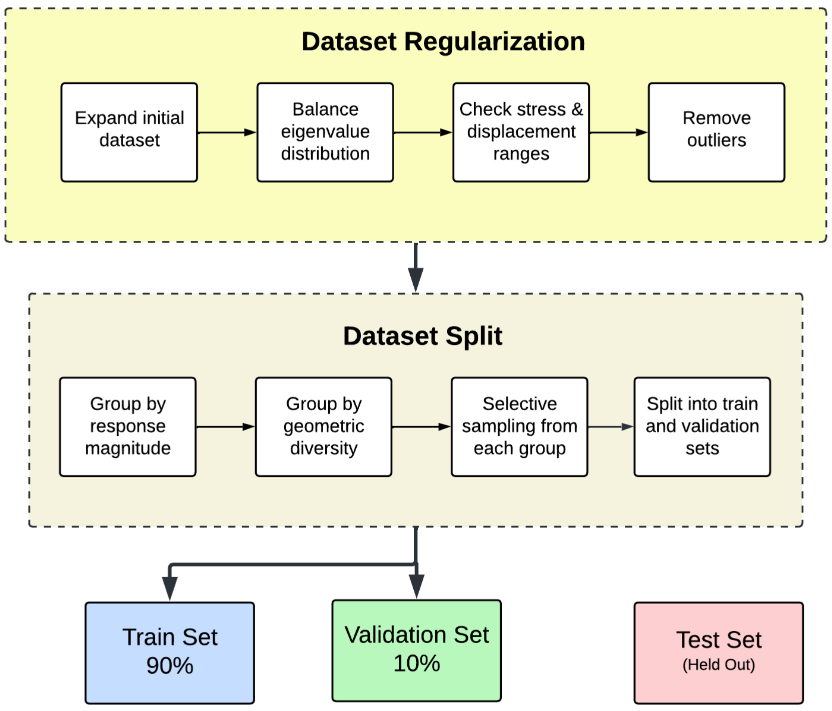

Dataset Regularization
Expand initial dataset
Balance eigenvalue distribution
Check stress & displacement ranges
Remove outliers
Dataset Split
Group by response magnitude
Group by geometric diversity
Selective sampling from each group
Split into train and validation sets
Train Set 90%
Validation Set 10%
Test Set (Held Out)

Figure 3.2: Flowchart for dataset regularization and processing

boundary points, where the number of points is randomly selected between 4 and 12 points, providing sufficient variability while maintaining reasonable geometric complexity. These boundary points are distributed around a base radius that ranges from 350 mm to 450 mm, with controlled variations to create natural, asymmetric forms. The algorithm incorporates several layers of controlled randomization to ensure geometric diversity:

- Angle Variation: Each boundary point's angular position includes a random variation of ±0.15 radians from its nominal position, creating natural irregularities in the shape outline.
- Radius Variation: The radius for each point varies between -50% to +40% from the base radius, with an additional sinusoidal variation (frequency multiplier of 3 and magnitude of 0.3) to create more complex contours.
- Inward Curves: With a 20% probability for each point, the algorithm applies an inward scaling factor between 0.4 and 0.7, creating concave regions in the shape.

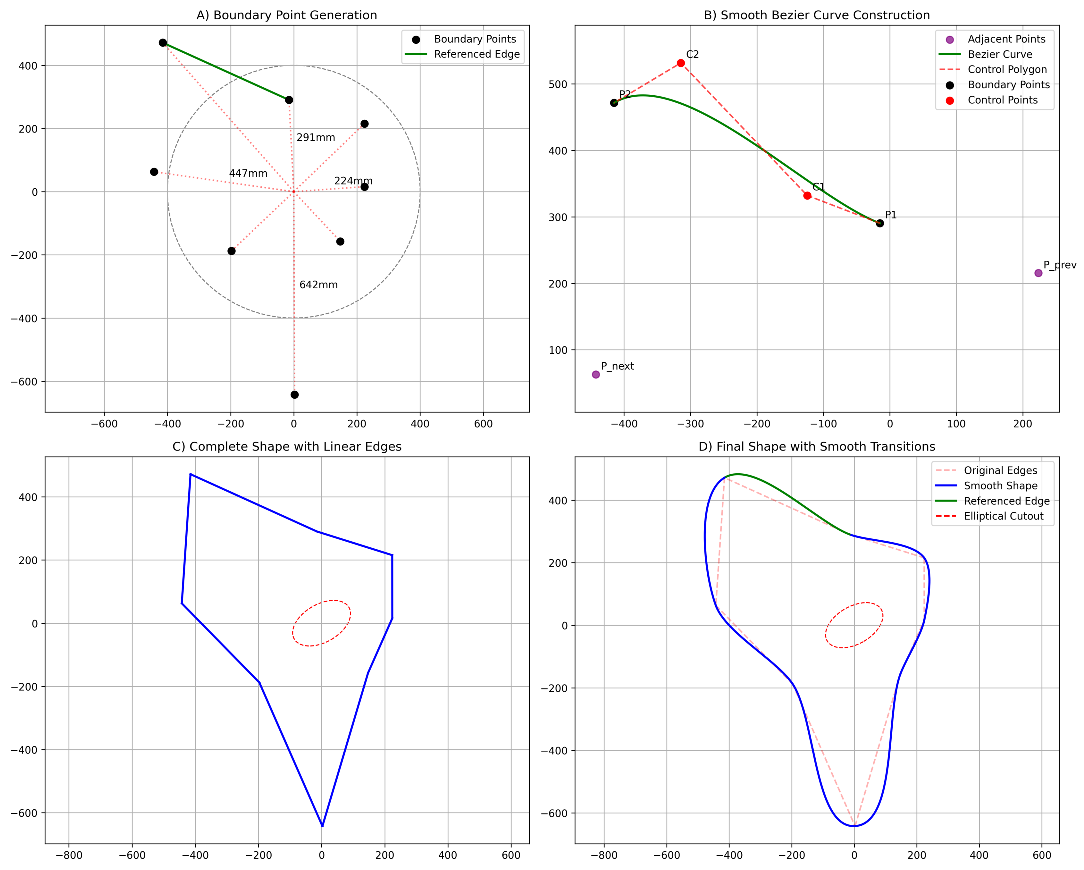


Figure 3.3: Shape generation process illustration

To ensure smooth transitions between boundary points, the algorithm employs a sophisticated control point generation system for Bezier curves. The control points are calculated using a minimum radius factor of 0.2 relative to the segment length, with up to 30% variation in control point length. This approach ensures both smoothness and variability in the resulting curves.

3.2.2 Shape Scaling and Validation

After the initial shape generation, a scaling process is applied to ensure all shapes fall within the specified dimensional constraints while maintaining their proportional characteristics. The scaling algorithm enforces the following criteria:

The scaling process involves; (1) calculating current bounds of the shape, (2) determining the maximum dimension, (3) computing the appropriate scale factor to fit

Table 3.1: Shape Generation Parameters

Parameter	Minimum Value	Maximum Value
Overall Size	700 mm	1000 mm
Number of Boundary Points	4	12
Angular Position Variation	-0.15 rad	+0.15 rad
Aspect Ratio	0.5	2.0
Base Radius	350 mm	450 mm
Radius Variation from Base	-50%	+40%
Inward Curve Probability	20%	20%
Inward Scaling Factor	0.4	0.7

within the target size range (700mm to 1000mm), (4) applying the scale factor while maintaining aspect ratio and (5) centering the shape at the origin.

Each generated shape undergoes validation to ensure it meets aspect ratio requirements (between 0.5 and 2.0) and maintains minimum feature sizes necessary for practical analysis.

3.2.3 Cutout Generation

For shapes designated to include cutouts, a sophisticated cutout generation system was implemented. The system can create both circular and elliptical cutouts, with the selection between these types governed by a probability parameter of 0.5. The cutout placement and sizing follow specific rules to ensure structural validity.

Table 3.2: Cutout Generation Parameters

Parameter	Minimum Value	Maximum Value
Cutout Size	60 mm	240 mm
Ellipse Aspect Ratio	0.5	1.5
Minimum Distance Factor	1.5	-
Maximum Cutouts per Shape	-	1

The cutout placement algorithm calculates safe interior regions using a minimum distance criterion from boundaries. Minimum spacing between cutouts and boundaries are selected to be 1.5 times the cutout diameter. For elliptical cutouts, the aspect ratio varies between 0.5 and 1.5, providing additional geometric diversity while maintaining manufacturability considerations.

Figure 3.4 visualize several generated shape examples according to explained shape generation algorithm. While left column of the figure present base shapes, right column provides example of shapes with cutout.

3.3 Finite Element Mesh Generation

To accurately capture the structural behavior best practices of meshing should be followed when creating the mesh. In the automobile, aerospace and many other industries where thin walled structures are commonly used, these structures are modelled with 2D shell elements because of their relatively small thickness. 2D shell elements represent the geometry of the structure very well for this kind of problems and also more efficient in terms of computational burden when compared to 3D elements. Therefore, for the base structure, we utilize 2D shell elements, specifically NASTRAN's CQUAD4 elements, with a quad-dominant meshing strategy. This choice aligns with established practices in thin-walled structural analysis, where the thickness dimension is significantly smaller than other geometric dimensions. Shell elements effectively capture both membrane and bending behaviors while maintaining computational efficiency compared to full 3D element formulations.

The meshing process is conducted using the commercial software HYPERMESH BATCHMESHER , which is a widely used tool for generating mesh in many industries. The meshing process involves the following steps:

1. Importing the generated shapes into HYPERMESH
2. Automatically meshing the shapes using the quad-dominant meshing strategy
3. Refining the mesh to ensure consistent element size and quality

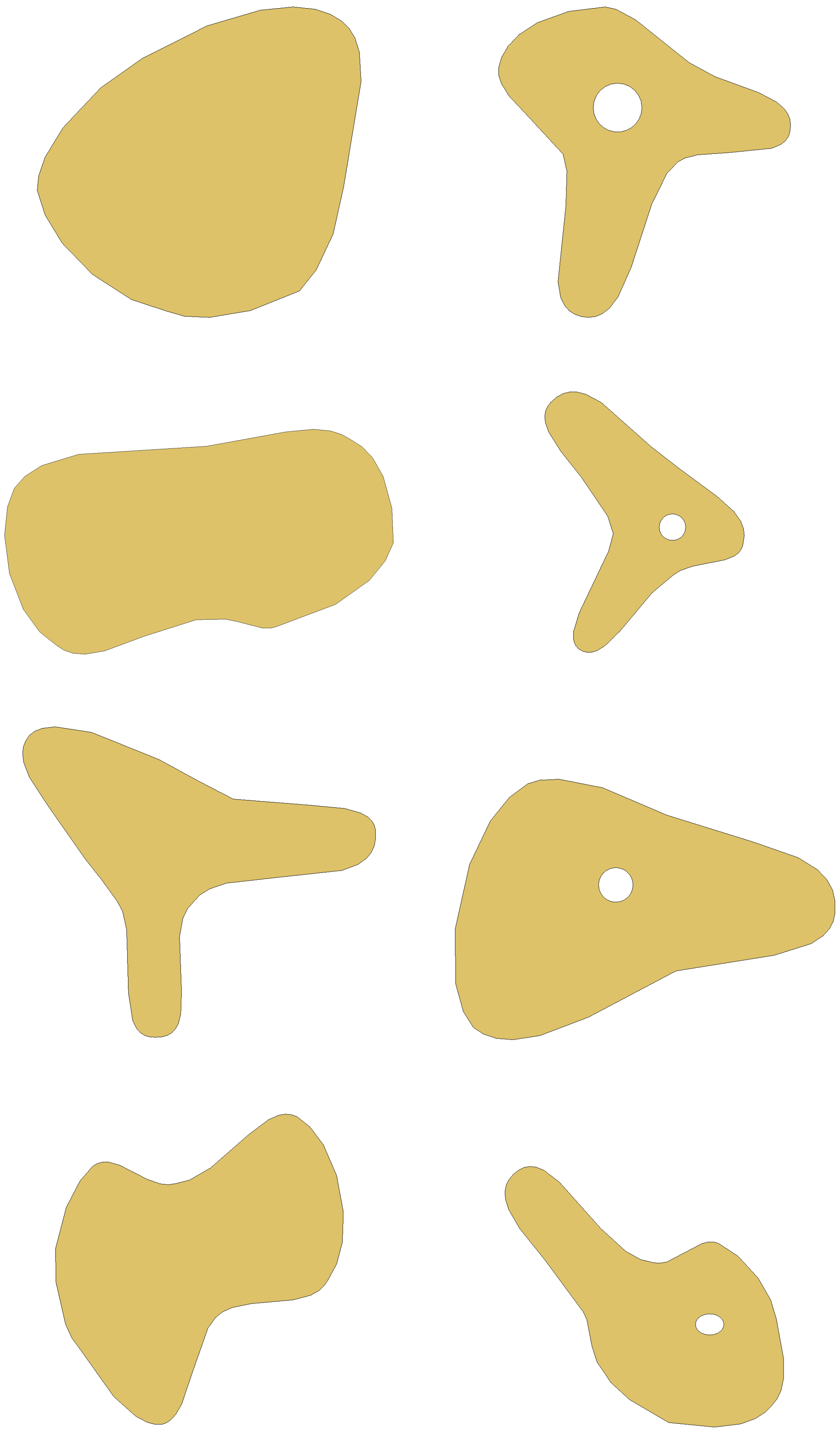


Figure 3.4: Examples of generated shapes

4. Exporting the mesh data in BDF format that can be used for the finite element analysis in NASTRAN

HYPERMESH uses a user defined quality criteria such as element aspect ratio, skew angle and jacobian to ensure the generated mesh meets the required quality standards. Applied quality criteria when creating the meshes is given in the Table 3.3.

Table 3.3: Mesh Generation Quality Criteria

Criterion	**Wt**	**Ideal**	**Good**	**Warn**	**Fail**
Min Length	2.0	22.00	16.00	12.00	8.00
Max Length	1.5	22.00	28.00	34.00	38.50
Aspect Ratio	1.0	1.00	2.00	4.00	5.00
Warpage	2.0	0.00	3.75	11.25	15.00
Max Angle Quad	1.0	90.00	110.00	140.00	150.00
Min Angle Quad	1.0	90.00	70.00	40.00	30.00
Max Angle Tria	1.0	60.00	80.00	120.00	130.00
Min Angle Tria	1.0	60.00	50.00	30.00	20.00
Skew	1.5	0.00	10.00	50.00	60.00
Jacobian	2.0	1.00	0.90	0.60	0.50
Chordal Dev	0.0	0.00	0.30	0.80	1.00
% of Trias	2.0	0.00	1.50	2.25	3.00
Taper	1.0	0.00	0.20	0.50	0.60
Penalty Value	-	0.00	0.00	1.00	2.00

Each mesh undergoes automated quality checking against the criteria defined in Table 3.3. Elements failing these criteria trigger automated mesh refinement and optimization procedures.

In regions containing cutouts, special attention is paid to mesh refinement and element transitions. The mesh density is not locally increased around cutout boundaries but washer meshing is implemented to accurately capture stress concentrations and potential local buckling modes. High element quality is achieved around cutout regions to minimize element based numerical artifacts.

The mesh generation process incorporates industry standard validation steps to ensure consistency across the entire dataset. This systematic approach to discretization provides the foundation for reliable finite element analysis and subsequent training of the graph neural network model.

Figure 3.5 illustrates results of mesh generation process for shapes given in Figure 3.4. Right column is belong to shapes with cutout that visualizes mesh refinement around cutouts. Left column of the figure provides base shape mesh generation.

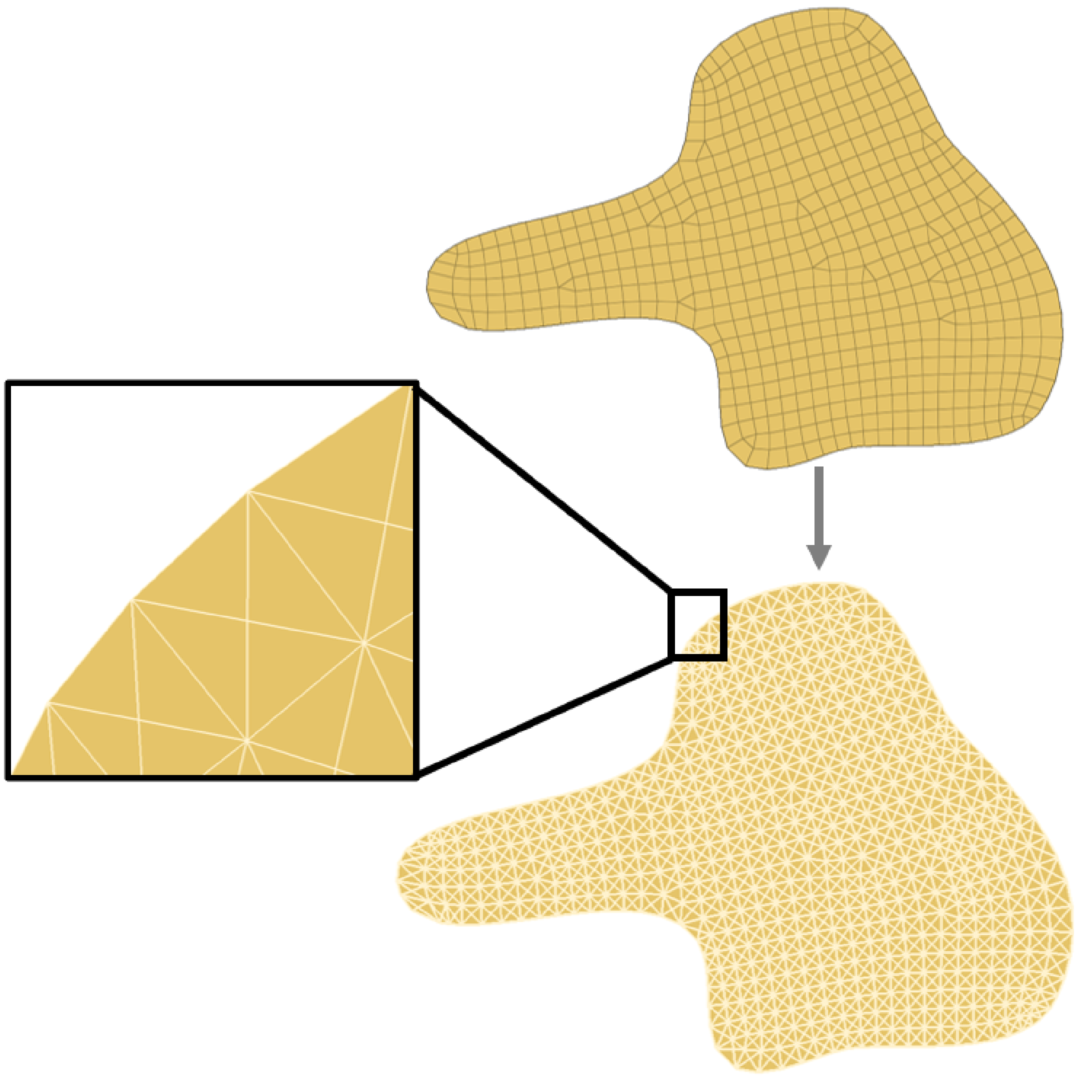


Figure 3.6: Ground stiffener network generation on a mesh

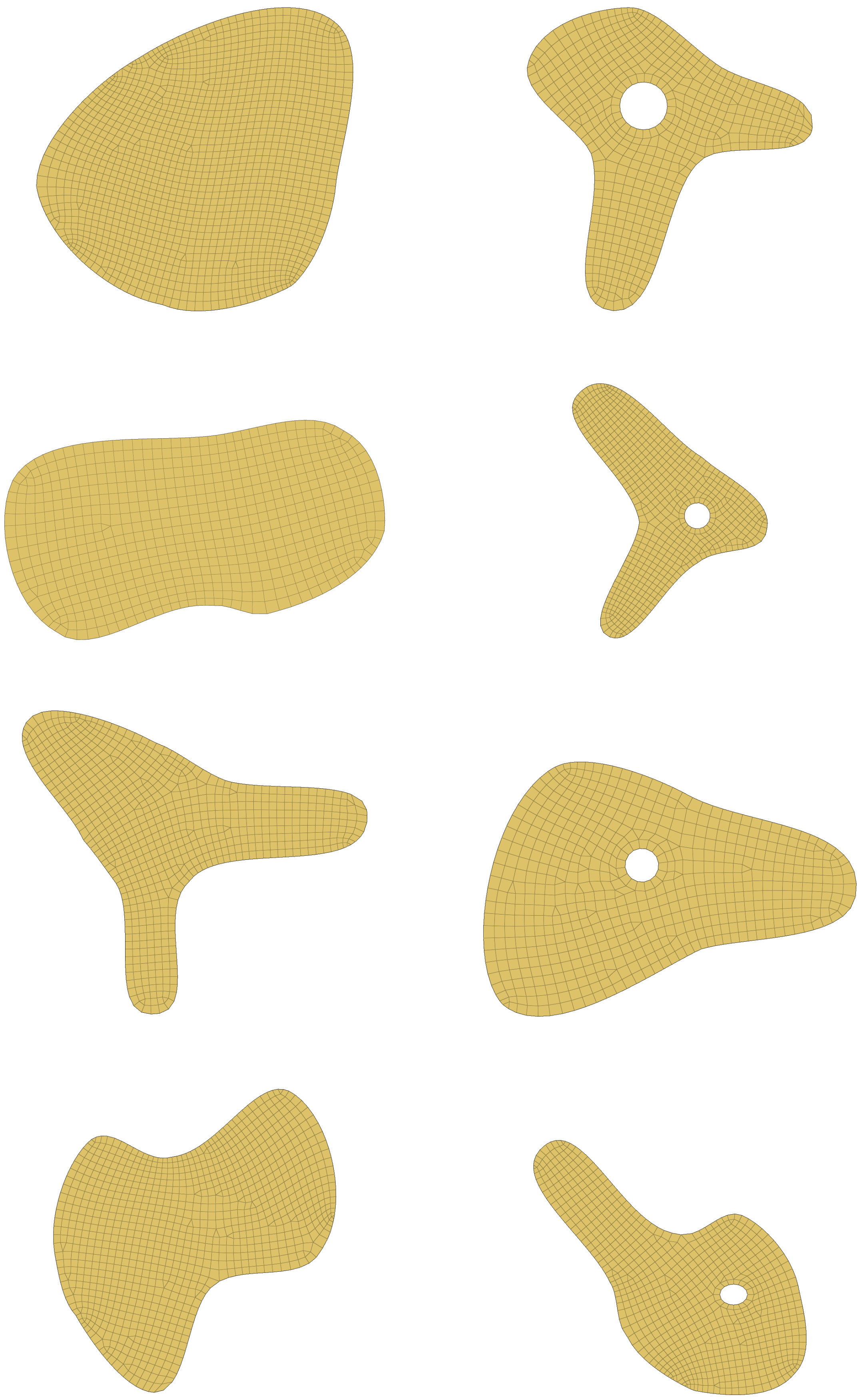


Figure 3.5: Examples of generated meshes

3.3.1 Stiffener Generation Strategy

For stiffened structures, we implement a systematic approach to stiffener placement and activation that balances structural effectiveness with computational feasibility. The generation process involves two key steps: first creating a network of potential stiffener locations which we name this "ground stiffener network" with dummy properties as shown in Figure 3.6, then selectively activating groups of these stiffeners to create meaningful structural configurations.

All potential stiffener locations are initially modeled using NASTRAN's CBAR elements, with two distinct property definitions: active stiffeners (PID=900) and inactive stiffeners (PID=999). The material properties for both use an aluminum alloy with Young's modulus E=76 GPa and Poisson's ratio ν=0.3 [13]. The active stiffeners with 2 mm thickness and 80 mm height are defined with the following cross-sectional properties.

- Area: 160 mm^2 (2.0 × 80.0 mm)
- Moment of inertia I$_1$: 85,333.33 mm^4 (2.0 × 80.0^3/12)
- Torsional constant J: 213.33 mm^4 (0.333 × 80.0 × 2.0^3)

Inactive stiffeners are assigned negligible property values (0.0001 for all parameters) to maintain mesh connectivity while minimizing their structural influence. This approach enables us to switch stiffeners between active and inactive states without modifying the underlying mesh topology. The stiffener activation process follows specific rules given in Table 3.4 to ensure realistic structural configurations.

Table 3.4: Stiffener Generation Parameters

Parameter	**Minimum Value**	**Maximum Value**
Active stiffeners	10	100
Consecutive stiffeners	5	25
Direction tolerance	0 deg	30 deg

The activation algorithm employs a group-based approach where stiffeners are activated in connected sequences. For each group:

1. A random starting edge is selected
2. Connected edges within the direction tolerance are identified
3. The group is grown bidirectionally until reaching either the maximum consecutive limit or finding no valid connections
4. The process repeats until reaching the target number of active stiffeners

This approach ensures structural continuity in stiffener patterns while maintaining sufficient variability in the dataset. The direction tolerance of 30 degrees ensures that activated stiffener groups follow reasonably straight paths, mimicking typical aerospace structural design practices.

1D stiffener layout example is given in Figure 3.7. In this figure red lines represent 1D stiffeners. All other transparent lines represent our ground stiffeners with dummy properties.

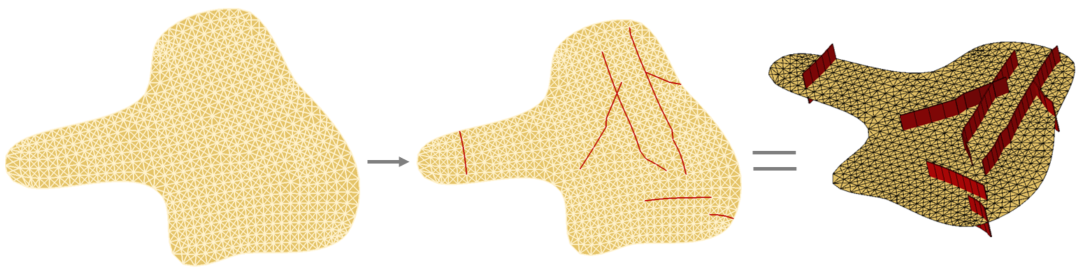


Figure 3.7: 1D stiffener generation process

This stiffener generation strategy was applied to create our enhanced dataset of stiffened structures, which ultimately comprised 714,704 unique configurations before the eigenvalue-based filtering process.

3.4 Load Case Generation

The generation of load cases represents a critical aspect of our methodology, as it directly influences the quality and diversity of the training dataset. We developed a systematic approach to create physically meaningful loading scenarios that challenge the structure in various ways while maintaining practical relevance.

3.4.1 Load Case Generation Strategy

Our approach to load case generation balances both the need for diverse loading conditions and the requirement for physically meaningful scenarios. Each shape receives multiple distinct load cases, with specific constraints on both the boundary conditions (BC) and applied forces.

The boundary conditions are limited to a maximum of 1 BC line per shape, where each line consists of 25 to 35 connected nodes along the structure's boundary. These nodes are constrained in all six degrees of freedom (three translations and three rotations) using NASTRAN's SPC1 card with a '123456' specification. This approach ensures proper structural support while avoiding over-constraint.

Table 3.5: Load Case Generation Parameters

Parameter	**Minimum**	**Maximum**
BC Line Node Count	25	35
Force Line Node Count	10	20
Force Magnitude (per node)	10 N	1000 N
Load Cases per Shape	10	10
BC Lines per Shape	1	1
Force Lines per Load Case	1	1

For force application, the algorithm generates one load line per load case, with each line containing 10 to 20 connected boundary nodes. The force magnitude on each node ranges from 10N to 1000N, providing sufficient variation in loading intensity.

The direction of the applied forces is determined through a random angle generation process that ensures coverage of various loading scenarios - from pure compression to combined compression-shear conditions.

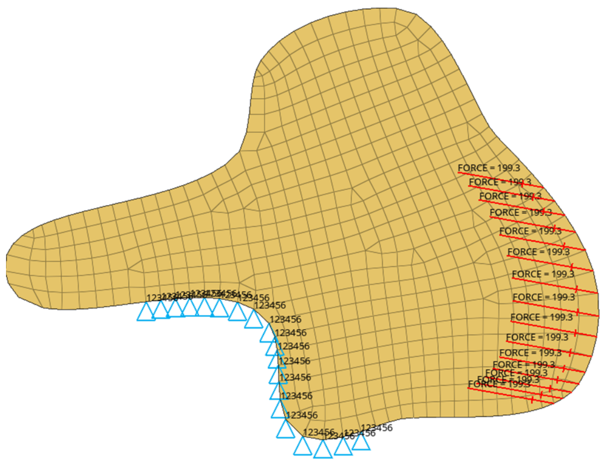


Figure 3.8: A generated load case example

3.4.2 Load Case Filtering and Validation

Each generated load case undergoes a validation process to ensure its suitability for training data. Load cases are checked to ensure that the maximum displacement remains within physically reasonable bounds, filtering out cases that might violate small displacement theory assumptions. Also buckling eigenvalue can be calculated negative even though positive eigenvalues request is explicitly defined in BDF files. These solutions are filtered out.

3.5 Finite Element Analysis

The finite element analysis phase of our study employs two distinct types of analyses: linear static analysis and linear buckling analysis. These analyses are conducted using MSC NASTRAN, chosen for its robust solution algorithms and wide acceptance in the engineering community. The analyses are performed sequentially, as the buckling analysis requires the stress state from the static solution to form the geometric stiffness matrix.

3.5.1 Linear Static Finite Element Analysis

Linear static analysis forms the foundation of our structural assessment, providing the stress state and deformation field under applied loads. The analysis assumes linear elastic material behavior and small deformation theory. The governing equation for this analysis is:

$$[K]u = F \tag{3.1}$$

where $[K]$ is the global stiffness matrix, u is the displacement vector, and F is the applied force vector. For our implementation, we utilize NASTRAN's Solution 101 [28] sequence with the following specifications:

- Material Properties: Aluminum alloy (E = 76 GPa, v = 0.3) [13]
- Element Formulation: CQUAD4 dominated elements for the base structure
- Output Requests: - DISPLACEMENT(PLOT) = ALL - STRESS(PLOT) = ALL - GPSTRESS(PLOT) = ALL

The analysis explicitly accounts for grid point stress balance and calculates surface stresses at grid points through the GPSTRESS command, which are crucial for our subsequent graph neural network implementation. The grid point stress output provides a more continuous stress field representation compared to element-centered corner stress results [28].

3.5.2 Linear Buckling Finite Element Analysis

Following the static analysis, we perform a linear buckling analysis to determine the critical buckling load and corresponding mode shape. As aforementioned, the eigenvalue problem solved in this analysis is:

$$([K] + \lambda[K_G])\phi = 0 \tag{3.2}$$

where $[K_G]$ is the geometric stiffness matrix derived from the static stress state, λ is the eigenvalue representing the buckling load factor, and ϕ is the eigenvector representing the buckling mode shape. Our implementation employs NASTRAN's Solution 105 [28] sequence with the following specifications:

- Method: Lanczos eigenvalue extraction (EIGRL card)
- Number of Eigenvalues: Limited to first positive eigenvalue for efficiency
- Stress-Based Differential Stiffness $[K_G]$: Automatically computed from Solution 101 results
- Output Requests: - DISPLACEMENT(PLOT) = ALL

The buckling analysis is set up to capture only the lowest eigenvalue, as this represents the critical buckling load factor that typically governs design. The analysis parameters are chosen to ensure computational efficiency considering large number of analyses required for our dataset.

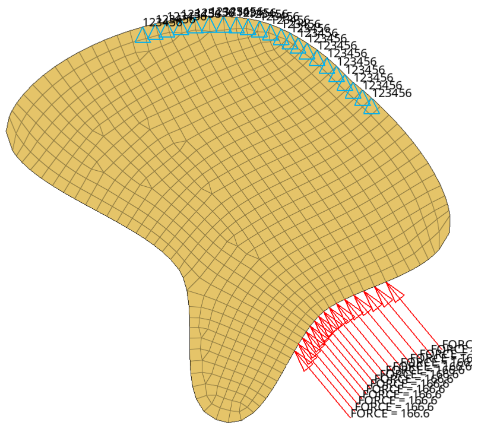
3.5.3 Output Processing and Validation

The analysis results undergo systematic processing and validation before being incorporated into the training dataset. Each analysis is checked for proper completion status. No hidden node should remain in any input file (.bdf format). Results are extracted from the output files (.op2 format) using specialized parsing routines that capture (1) nodal displacements and rotations, (2) grid point stress components (σ_x, σ_y, τ_{xy}), (3) buckling eigenvalue and mode shape. Results are validated against maximum displacement magnitude and eigenvalue positivity check. The maximum stress is arranged so that the maximum stress in the model does not exceed 300MPa.

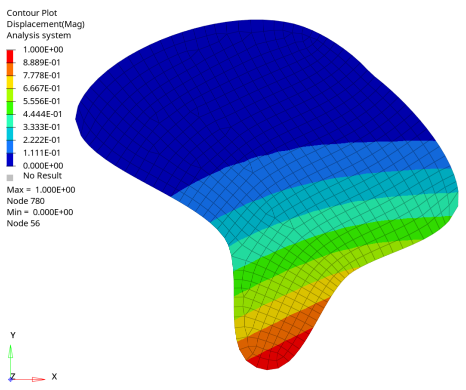
The finite element analysis phase provides the foundation for training our graph neural network model. The careful consideration of analysis parameters and validation criteria ensures the reliability of our training data, while the comprehensive output processing enables effective translation of finite element results into the graph-based format required for machine learning implementation.

3.6 Dataset Regularization

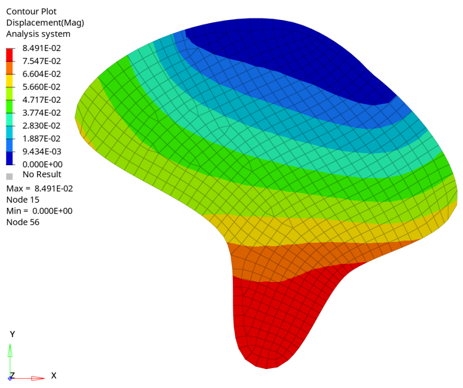
The development of an effective surrogate model requires not only accurate data but also a well-balanced dataset that avoids biases in the training process. Initial analysis



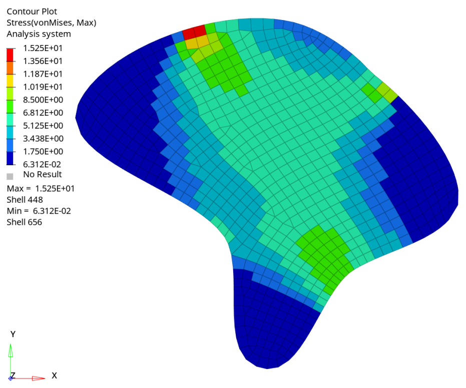
(a) Generated load case



(b) Buckling mode shape plot with 0.09 eigenvalue



(c) Displacement magnitude plot



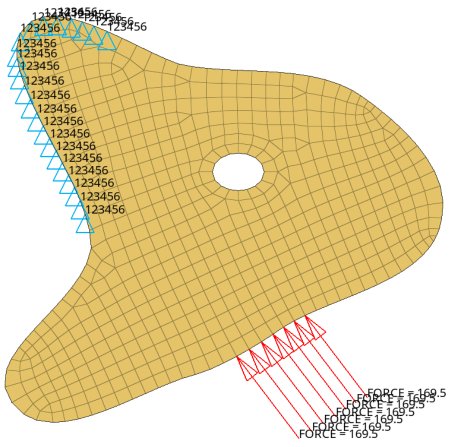
(d) Von Mises stress field plot

Figure 3.9: Examples of finite element analysis results of a shape without cutout

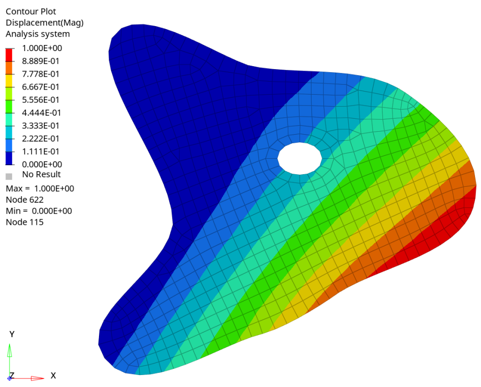
of our generated datasets revealed significant imbalances in the distribution of structural responses, particularly in buckling eigenvalues, which could potentially bias the model training [54]. This section describes our systematic approach to dataset regularization, focusing on achieving a more uniform distribution of critical structural responses while maintaining the physical validity of the data.

3.6.1 Dataset without Stiffeners

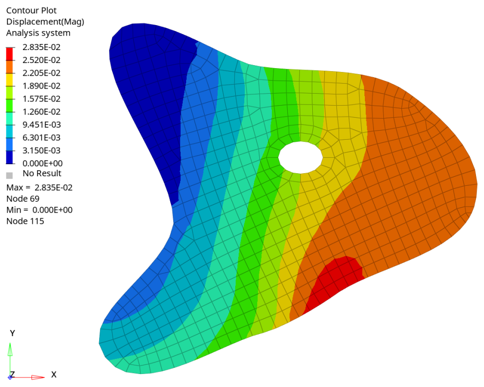
Our initial dataset without stiffeners, comprising 40,000 analysis cases (4,000 shapes with 10 load cases each), exhibited a highly skewed distribution of buckling eigenval-



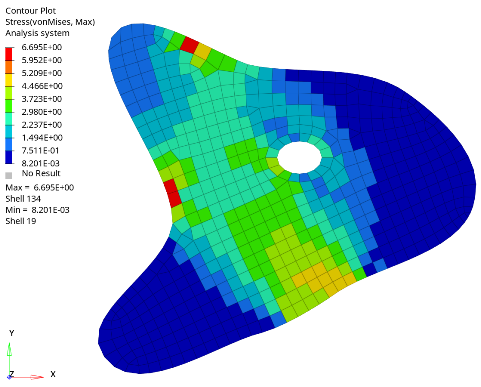
(a) Generated load case



(b) Buckling mode shape plot with 0.06 eigenvalue



(c) Displacement magnitude plot



(d) Von Mises stress field plot

Figure 3.10: Examples of finite element analysis results of a shape with cutout

ues. As shown in Figure 3.11, the distribution followed an approximately logarithmic pattern, with a dense concentration of cases at lower eigenvalues and a long tail extending toward higher values. This distribution pattern is inherent to the physics of buckling behavior, where geometric variations and loading conditions tend to produce more cases with lower critical loads.

3.6.1.1 Dataset Expansion and Under-sampling

To address this distributional bias, we implemented a two-phase approach. First, we expanded the dataset by a factor of 5, generating approximately 200,000 analysis

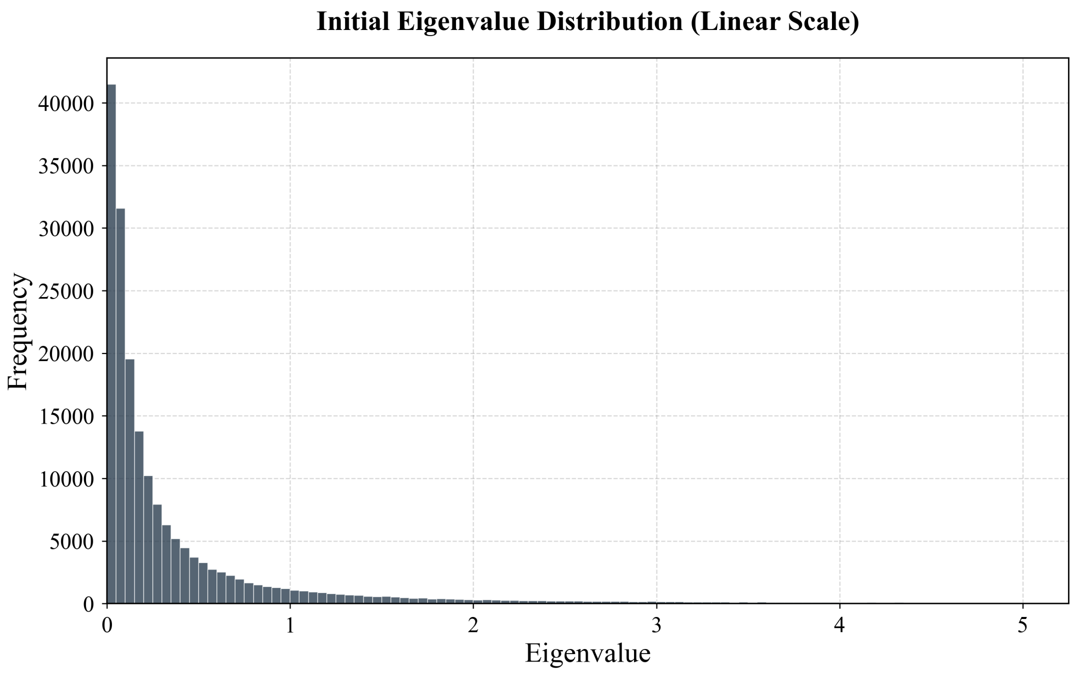


Figure 3.11: Initial distribution of buckling eigenvalues for non-stiffened dataset

cases. This expansion maintained the same geometric and loading parameters but provided a larger pool for selective sampling. The expanded dataset provided more data in regions with higher eigenvalues, allowing us to achieve better representation across the entire eigenvalue range. Second, we applied selective sampling to achieve a more uniform distribution. The eigenvalue range was divided into bins of equal width of 0.05, and a target count was established for each bin based on the desired uniform distribution. Cases were selectively sampled from each bin to achieve the target distribution, with bins exceeding the target count being subsampled, while maintaining diversity in geometric and loading conditions.

The regularization process resulted in a final dataset of 40,000 cases with approximately uniform distribution of buckling eigenvalues across the retained range. The initial distribution and the final distribution are illustrated in Figure 3.12. This balanced dataset forms the foundation for training our graph neural network model, ensuring that the model receives sufficient examples across the entire range of buckling behaviors. With using final distribution shown in Figure 3.12, eigenvalue imbalance is handled as each eigenvalue range is represented in training.

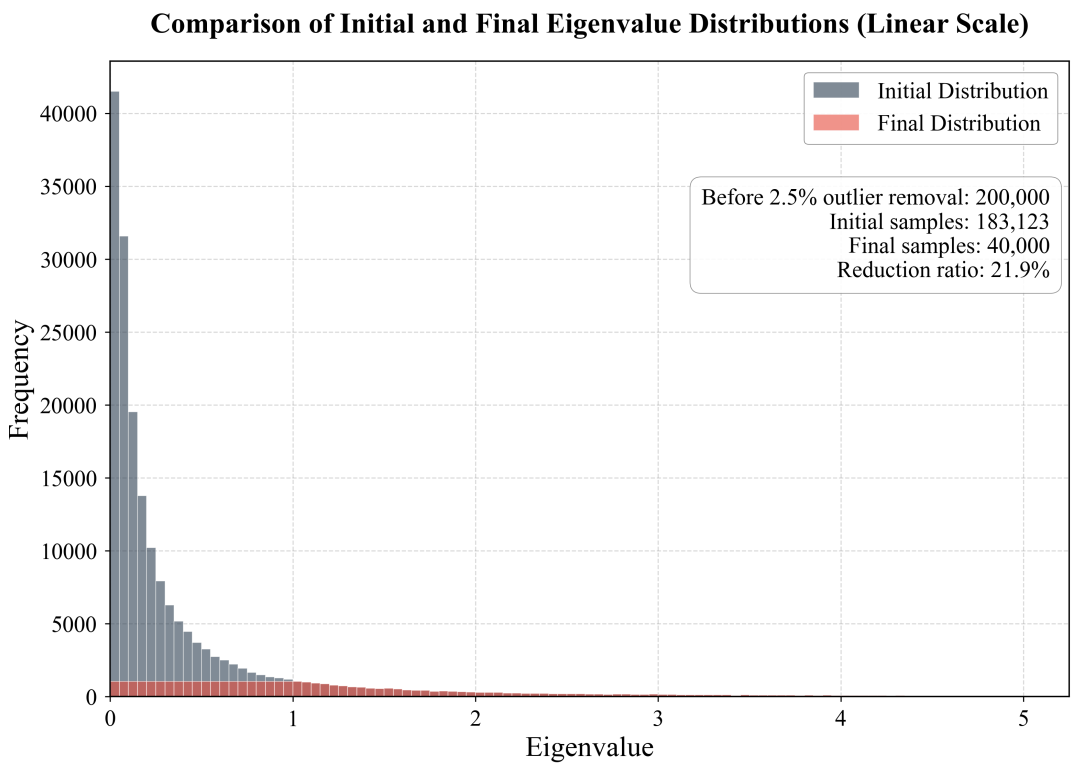


Figure 3.12: Dataset regularization process for non-stiffened dataset

3.6.2 Enhanced Dataset with Stiffeners

The regularization process for the stiffened structure dataset followed a similar methodology but with a larger initial dataset owing to the increased complexity of stiffened configurations. Starting with 714,704 analysis cases, we first removed cases exhibiting numerically unstable or physically unrealistic behaviors, resulting in 678,968 valid cases. This filtering step was crucial for ensuring the quality of our training data.

After the initial filtering, we applied a similar binning and selective sampling approach as used for the non-stiffened dataset. However, the larger initial dataset allowed us to maintain a higher number of samples per bin while still achieving a relatively uniform distribution. The final regulated dataset consisted of 80,000 cases, representing approximately 11.8% of the initial valid cases. Figure 3.13 illustrates this dramatic transformation from the highly skewed initial distribution to a more balanced final distribution.

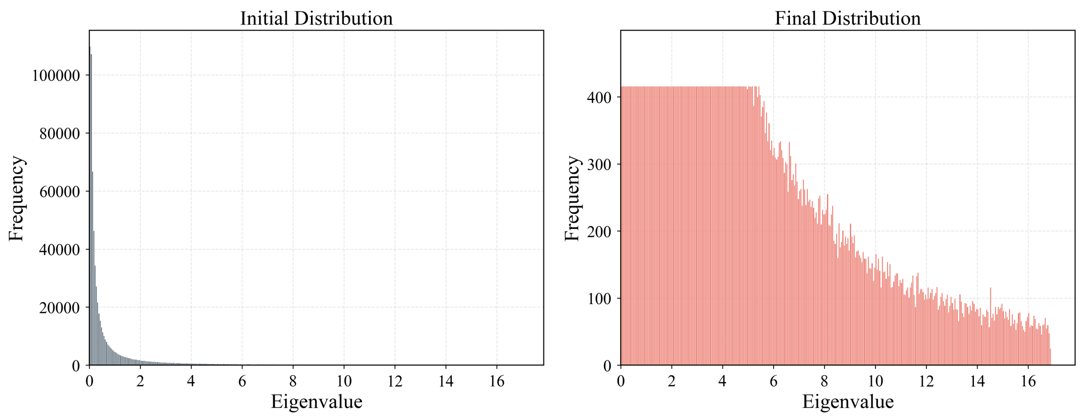


Figure 3.13: Comparison of initial and final eigenvalue distributions for stiffened dataset

The regularization process for the stiffened dataset presented additional challenges due to the complex interactions between stiffeners and the base structure. These interactions resulted in a wider range of eigenvalues and more intricate patterns in their distribution. Despite these challenges, our approach successfully achieved a relatively uniform distribution while preserving the diversity of structural configurations and loading conditions.

3.6.3 Impact on Static Displacement Dataset

In contrast to the buckling analysis datasets, we did not apply similar regularization procedures to the static displacement dataset. This decision was primarily driven by two factors. First, displacement fields represent local structural responses rather than global properties like buckling eigenvalues, making their distribution patterns fundamentally different. Second, our focus on buckling prediction as the primary objective led us to prioritize the quality and balance of buckling-related data.

The lack of regularization for the static displacement dataset likely contributed to the reduced performance we observed in displacement field predictions compared to the results reported by Gladstone et al. [11] in their EA-GNN implementation. As identified by Yang et al. [54], regression tasks can suffer from data imbalance where certain value ranges are underrepresented, leading to biased models. This phenomenon likely

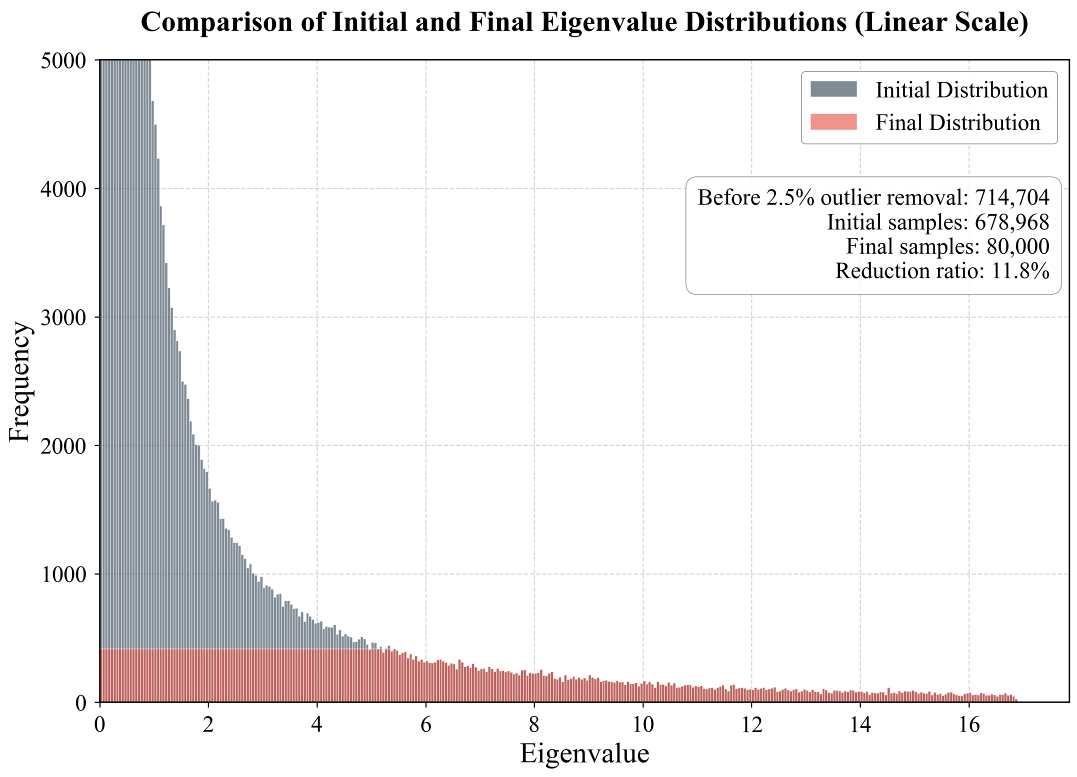


Figure 3.14: Dataset regularization process for stiffened dataset

explains the performance gap between our work and Gladstone's results.

3.7 Dataset Split

The effective partitioning of data into training, validation, and test sets represents an important step in developing robust machine learning models. Our approach to dataset splitting emphasizes not only the statistical distribution of target values but also the preservation of geometric and loading diversity across all sets. This section details our comprehensive splitting methodology and the creation of specialized test datasets for evaluating model generalization.

3.7.1 Training and Validation Split Strategy

For both non-stiffened and stiffened datasets, we implemented a splitting strategy to maintain balanced representation across different structural behaviors. The non-stiffened dataset of 40,000 cases and the stiffened dataset of 80,000 cases were split using a 90-10 ratio for training and validation sets respectively. The splitting process was designed to preserve the balanced eigenvalue distribution achieved through our regularization process in both sets, ensuring reliable model evaluation across the entire spectrum of structural responses.

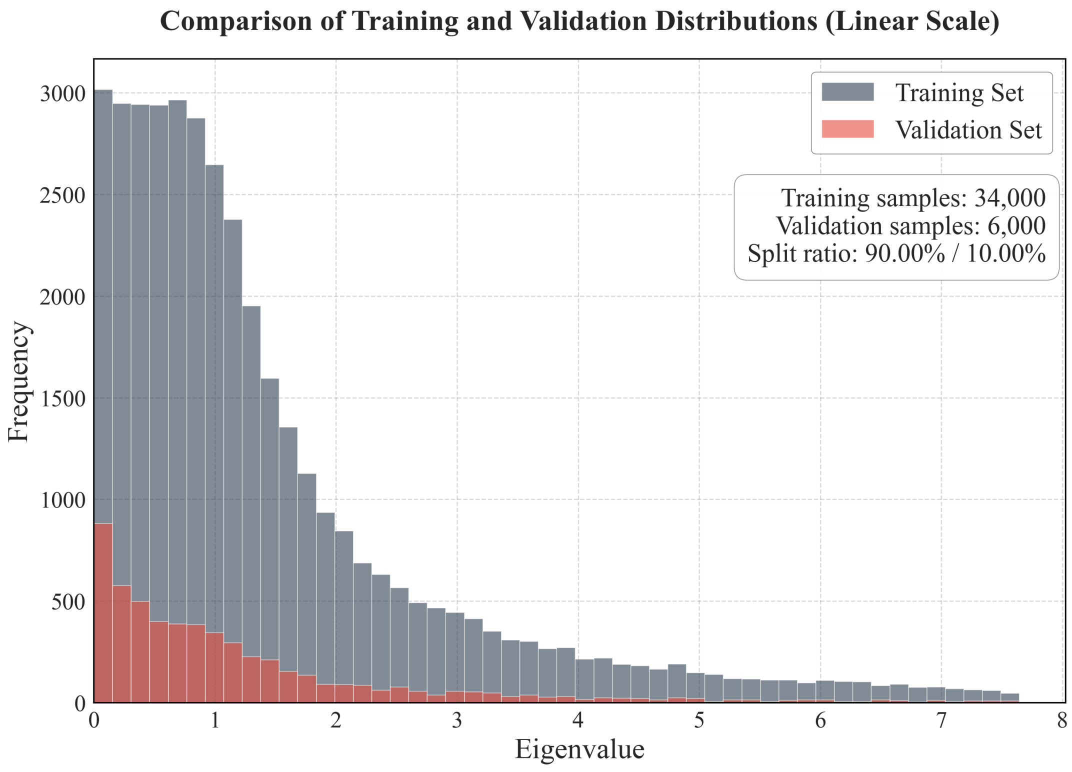


Figure 3.15: Distribution analysis of training and validation sets for non-stiffened dataset

As illustrated in Figure 3.15 and 3.16, both training and validation sets maintain similar distribution patterns, with each eigenvalue bin represented proportionally. This balanced representation is crucial for reliable model evaluation, as it ensures that validation performance metrics reflect the model's capabilities across the entire spectrum of structural behaviors.

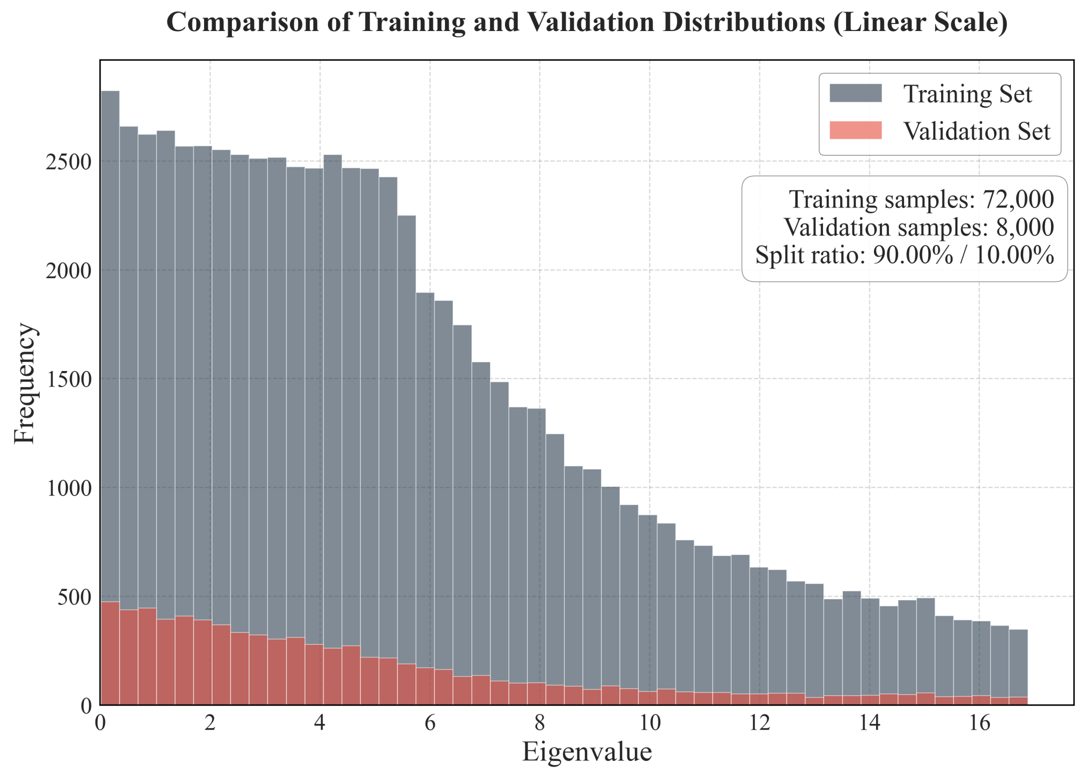


Figure 3.16: Distribution analysis of training and validation sets for stiffened dataset

3.7.2 Test Dataset Generation

To comprehensively evaluate model generalization, we developed multiple specialized test datasets for both non-stiffened and stiffened configurations. These test sets were designed to assess different aspects of model performance and robustness.

3.7.2.1 Non-Stiffened Test Datasets

The base test dataset for non-stiffened structures consists of 4,000 analysis cases derived from 400 unique shapes not used in training or validation. While these shapes were generated using the same parameters as the training dataset, they represent entirely new geometries, ensuring a true test of the model's ability to generalize to unseen node configurations.

To evaluate the model's ability to handle geometric scaling, we created additional test datasets by scaling these 400 base test shapes. Additionally, we created test cases with modified loading configurations, incorporating multiple line loading and multiple SPC lines to assess the model's robustness to variations in loading patterns.

3.7.2.2 Stiffened Test Datasets

For stiffened structures, we developed a similar test dataset using the same 400 base shapes. The test configuration was modified to explore a broader range of stiffener layouts. 4 load cases per shape were selected and different stiffener layouts were generated per load case. This resulted in 8,000 total test cases (400 shapes × 4 load cases × 5 layouts). The test set represents approximately 10% of the stiffened training dataset size.

This configuration allows us to evaluate the model's performance across various stiffener patterns while maintaining consistent base geometry and loading conditions. The multiple stiffener layouts per load case enable assessment of the model's sensitivity to changes in stiffener configuration.

Table 3.6: Test Dataset Characteristics

Test Dataset	**Shapes**	**Cases/Shape**	**Layouts/Case**	**Total Cases**
Non-Stiffened Base	400	10	1	4,000
Non-Stiffened 0.5× Scale	400	10	1	4,000
Non-Stiffened 2.0× Scale	400	10	1	4,000
Non-Stiffened Multiple Loading	400	10	1	4,000
Stiffened Base	400	4	5	8,000
Stiffened 0.5× Scale	400	4	5	8,000
Stiffened 2.0× Scale	400	4	5	8,000
Stiffened Multiple Loading	400	4	5	8,000

The comprehensive nature of these test datasets enables thorough evaluation of model performance across different scenarios. This systematic approach to dataset splitting and test set generation provides a robust framework for evaluating model performance and generalization capabilities. Consideration of various geometric configurations, loading conditions, and stiffener patterns in our test datasets enables thorough assessment of the model’s practical applicability.

CHAPTER 4

GRAPH CREATION AND GNN MODEL ARCHITECTURE

4.1 Introduction

This chapter presents the comprehensive architecture and implementation details of our Graph Neural Network (GNN) framework for structural analysis. The development of this framework required careful consideration of several key aspects: the representation of structural data as graphs, enhancement of information flow between nodes, coordinate frame normalization for improved generalization and the design of specialized neural network architectures for different prediction tasks.

Our framework employs GraphSAGE as the primary architecture, chosen for its proven effectiveness in inductive learning tasks and its ability to generate node embeddings through localized neighbor sampling and aggregation [20]. This choice was motivated by GraphSAGE's demonstrated capability to learn on previously unseen graph structures, a crucial requirement for our structural analysis applications where each new geometry presents a unique graph topology. The framework incorporates several architectural innovations to address the specific challenges of structural analysis. First, we introduce enhanced information flow mechanisms through random virtual edges and a super node architecture, designed to facilitate both local and global information propagation across the structure. Second, we implement a coordinate frame normalization strategy based on Principal Component Analysis (PCA) to ensure rotational and translational invariance of our predictions. Finally, we develop specialized pooling strategies optimized for different prediction tasks, particularly focusing on the global nature of buckling eigenvalue prediction.

Our implementation leverages the PyTorch framework with PyTorch Geometric extensions, enabling efficient processing of graph-structured data while maintaining flexibility for custom architectural components. The framework is designed to handle both simple plate structures and more complex configurations with stiffeners.

The following sections provide detailed descriptions of each component, including mathematical formulations, implementation details and design rationales. Particular attention is paid to the architectural choices that enable effective learning of both local and global structural behaviors, which is crucial for accurate prediction of buckling eigenvalues and displacement fields.

4.2 Graph Representation of Structural Data

The transformation of structural analysis data into graph representations forms the foundation of our framework. This section details our comprehensive approach to encoding both geometric and physical properties of structures into a graph format suitable for neural network processing.

4.2.1 Node Feature Engineering

Node features in our framework are designed to capture both geometric and physical properties of the structural system. The feature vector for each node comprises several key components, engineered specifically for different prediction tasks. Demonstration of node feature engineering is illustrated in Figure 4.1.

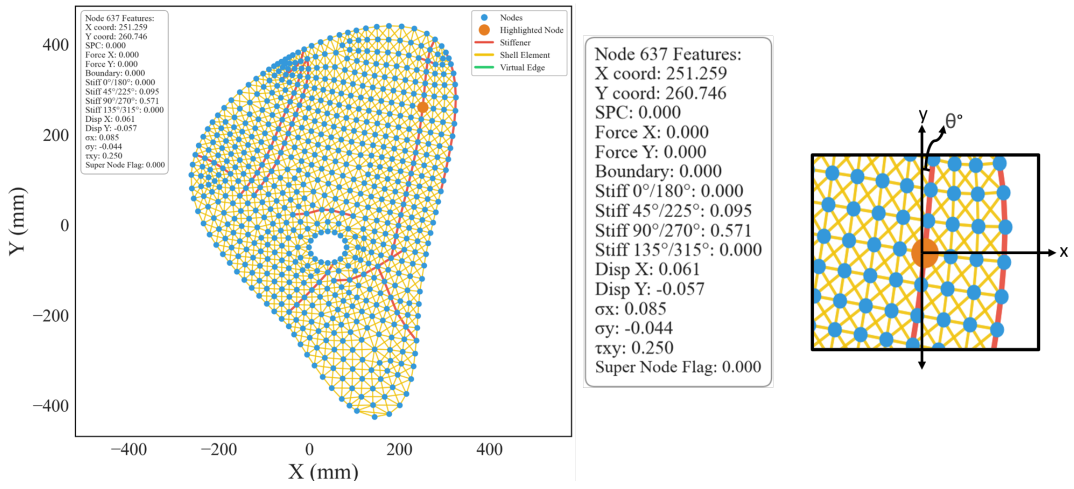
For static analysis prediction types (like displacements and stresses), nodes incorporate the following geometric features:

$$\mathbf{x}_{\mathbf{static}} = [\mathbf{p}, b, \mathbf{f}, o, \mathbf{s}] \tag{4.1}$$

where:

- $\mathbf{p} \in \mathbb{R}^2$ represents spatial coordinates $[x, y]$
- $b \in 0, 1$ is the fixed boundary condition flag, if node is constrained, this value is set to 1

- $\mathbf{f} \in \mathbb{R}^2$ represents external force components $[F_x, F_y]$
- $o \in 0, 1$ encodes boundary information, if node is at the boundary of the structure then this value is set to 1
- $\mathbf{s} \in \mathbb{R}^4$ stiffener information



Node 637 Features:
X coord: 251.259
Y coord: 260.746
SPC: 0.000
Force X: 0.000
Force Y: 0.000
Boundary: 0.000
Stiff 0°/180°: 0.000
Stiff 45°/225°: 0.095
Stiff 90°/270°: 0.571
Stiff 135°/315°: 0.000
Disp X: 0.061
Disp Y: -0.057
σx: 0.085
σy: -0.044
τxy: 0.250
Super Node Flag: 0.000

Figure 4.1: Demonstration of node feature engineering

For structures with stiffeners, the stiffener contribution at a particular node is quantified using directional bins:

$$\mathbf{s} = \left[s_{0/180}, s_{45/225}, s_{90/270}, s_{135/315}\right] \tag{4.2}$$

where each bin s_θ represents the normalized stiffener contribution in the corresponding direction range. These bins are calculated based on the geometric configuration of connected stiffener elements, with values normalized to the range [0, 1] by dividing by 3.0 as specified in our implementation. For example, if there is a node connecting two 1D stiffener elements which both lie on 0° direction then this particular node's $s_{0/180}$ value is $2/3 = 0.666$.

For buckling analysis, the node feature vector is extended to include:

$$\mathbf{x_{buckling}} = \left[\mathbf{x_{static}}, \mathbf{u}, \boldsymbol{\sigma}\right] \tag{4.3}$$

where:

- $\mathbf{u} \in \mathbb{R}^2$ represents static displacements (u_x, u_y)

- $\boldsymbol{\sigma} \in \mathbb{R}^3$ represents stress components (σ_x, σ_y, τ_{xy})

When employing the super node architecture, we append an additional binary indicator to distinguish between regular nodes and the super node. According to the prediction type that we use, final node features becomes:

$$\mathbf{x_{final\_static}} = [\mathbf{x}_{static}, n_{type}] \tag{4.4}$$

$$\mathbf{x_{final\_buckling}} = [\mathbf{x}_{buckling}, n_{type}] \tag{4.5}$$

where n_{type} is 0 for regular nodes and 1 for the super node.

4.2.2 Edge Feature Engineering

Edge features in our framework are designed to capture both topological connectivity and geometric relationships between nodes. We define three distinct types of edges: mesh edges, stiffener edges and virtual edges, each carrying specific attribute information. The process is visualised in Figure 4.2. For edges representing the finite element mesh connectivity, we encode both geometric and physical properties in a feature vector:

$$\mathbf{e_{mesh}} = [t, l_{norm}, \mathbf{d}, v] \tag{4.6}$$

where:

- $t \in 0, 1$ is stiffener flag, if there is a 1D stiffener on an edge then it is 1 for that particular edge
- l_{norm} is the normalized edge length (divided by 1000 for numerical stability)
- $\mathbf{d} \in \mathbb{R}^2$ is the normalized direction vector $[\mathbf{d_x}, \mathbf{d_y}]$
- $v \in 0, 1$ is the virtual edge flag, if edge is virtual then it is 1 for that particular edge

For stiffener elements, the first component (1.0) distinguishes stiffener edges from shell mesh edges. The increased magnitude of this component (1.0 vs 0.0) reflects the enhanced stiffness contribution of stiffener elements compared to regular mesh connections.

The virtual edge indicator ($v = 1$) explicitly marks virtual edges as non-physical connections. These edges are created according to the algorithm described in Section 4.3.1, with their number limited to 20.0% of the original edge count to maintain computational efficiency while ensuring enhanced information flow.

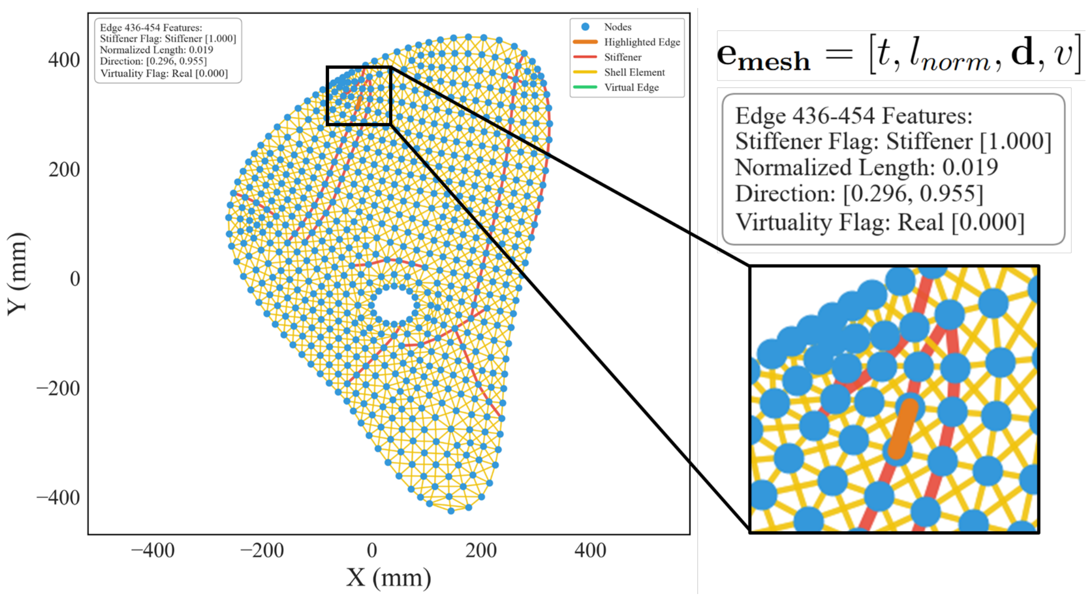


Figure 4.2: Demonstration of edge feature engineering

4.2.3 Feature Normalization

Our framework employs a comprehensive normalization strategy to ensure stable training and effective learning. Different feature types undergo specific normalization procedures. Edge features are already normalized when they are extracted. Initial node features are given in 4.7 and 4.8.

$$\mathbf{x_{displacement}} = [\mathbf{p}, b, \mathbf{f}, o, \mathbf{s}, n_{type}] \tag{4.7}$$

$$\mathbf{x_{buckling}} = [\mathbf{p}, b, \mathbf{f}, o, \mathbf{s}, \mathbf{u}, \boldsymbol{\sigma}, n_{type}] \tag{4.8}$$

Spatial coordinates are normalized using the global maximum extent of the structure,

$$\mathbf{p_{norm}} = \frac{\mathbf{p}}{\max(|p_{max}|, |p_{min}|)} \tag{4.9}$$

where p_{max} and p_{min} represent the maximum and minimum dimensions across all coordinates.

External forces are normalized based on the global force statistics,

$$\mathbf{f}_{norm} = \frac{\mathbf{f}}{\max(|\mathbf{f}_{max}|, |\mathbf{f}_{min}|)} \tag{4.10}$$

where f_{max} and f_{min} represent the maximum and minimum forces in x and y direction.

For displacements and stresses we employ robust scaling, specifically ScikitLearn's RobustScaler [55].

$$\theta_{\text{scaled}} = \frac{\theta - \mu}{\omega} \tag{4.11}$$

where μ and ω are computed using the RobustScaler implementation. The center of data is denoted as μ, the size factor is denoted as ω. RobustScaler makes the normalization resistant to outliers. We maintain separate scalers for different feature types and ensure consistent normalization across training and inference.

4.3 Enhanced Information Flow Mechanisms

A key challenge in applying GNNs to structural analysis is ensuring effective information propagation across the entire structure. This is particularly crucial for predicting global behaviors such as buckling modes, where local deformations can significantly influence the overall structural response. To address this challenge, we implement two complementary mechanisms: random virtual edge creation and a super node architecture.

4.3.1 Random Virtual Edge Creation

Random virtual edges are additional connections in the graph that supplement the physical mesh connectivity, enabling more direct information flow between distant regions of the structure. Figure 4.3 illustrates virtual edges drawn with red-dotted lines in Figure 4.3b. Figure 4.3a shows base graph without any virtual edge connection. The percentage of virtual edges (20%) was determined through empirical testing to balance enhanced connectivity with computational efficiency by Gladstone et al.

[11]. Random virtual edge creation process follows a controlled approach given in Algorithm 1.

Algorithm 1 Random Virtual Edge Generation

```
  Calculate maximum allowed virtual edges: $N_{virtual} = 20.0\% \times E_{mesh}$
  Initialize empty set $E_{virtual}$
  **while** $|E_{virtual}| < N_{virtual}$ **do**
    Select random nodes $(n_i, n_j)$ where $i \neq j$
    **if** $(n_i, n_j) \notin E_{mesh} \cup E_{virtual}$ **then**
      Add $(n_i, n_j)$ to $E_{virtual}$
    **end if**
  **end while**
```

Virtual edges are assigned following the format,

$$\mathbf{e}_v = [0, dij/1000, \mathbf{u}_{ij}, 1] \tag{4.12}$$

where d_{ij} is the Euclidean distance between nodes i and j $\mathbf{u}_{ij}$ is the unit direction vector and the last component (1) identifies the edge as virtual.

4.3.2 Super Node Architecture

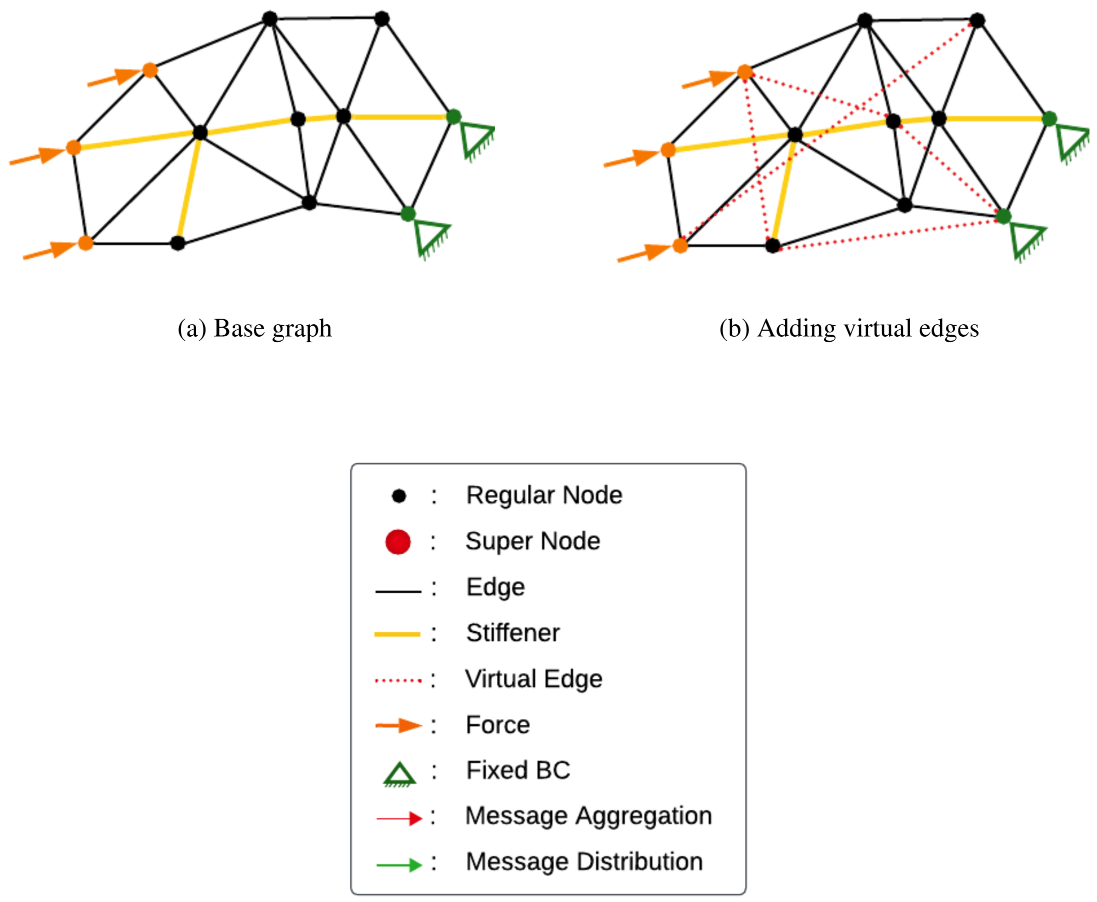
The super node architecture introduces a special node that connects to all other nodes in the graph, serving as a global information aggregator and distributor. The super node is implemented with the node feature vector given in 4.13.

$$\mathbf{x}_s = [\mathbf{0}, ..., \mathbf{0}, 1] \tag{4.13}$$

where the final component (1) identifies it as a super node. The super node connects to all regular nodes through bidirectional edges.

$$E_s = (v_s, v_i) \cup (v_i, v_s) | \forall v_i \in V \tag{4.14}$$

Finally, super node connections (edges) use the same feature format as virtual edges but maintain their distinct role through the graph's adjacency structure.



(a) Base graph

(b) Adding virtual edges

Figure 4.3: Virtual edge enhancement of graph connectivity

4.3.2.1 Information Flow Enhancement

The super node architecture provides several key benefits. Firstly, it reduces the maximum path length between any two nodes to 2, enabling faster information propagation.

$$d_{max} = 2 \forall v_i, v_j \in V \tag{4.15}$$

Super node utilization enables each node to access global structural information through the super node's aggregated features. Super node also facilitates both local and global feature processing through the dual-path information flow. Super node architecture and information flow is illustrated in Figure 4.6.

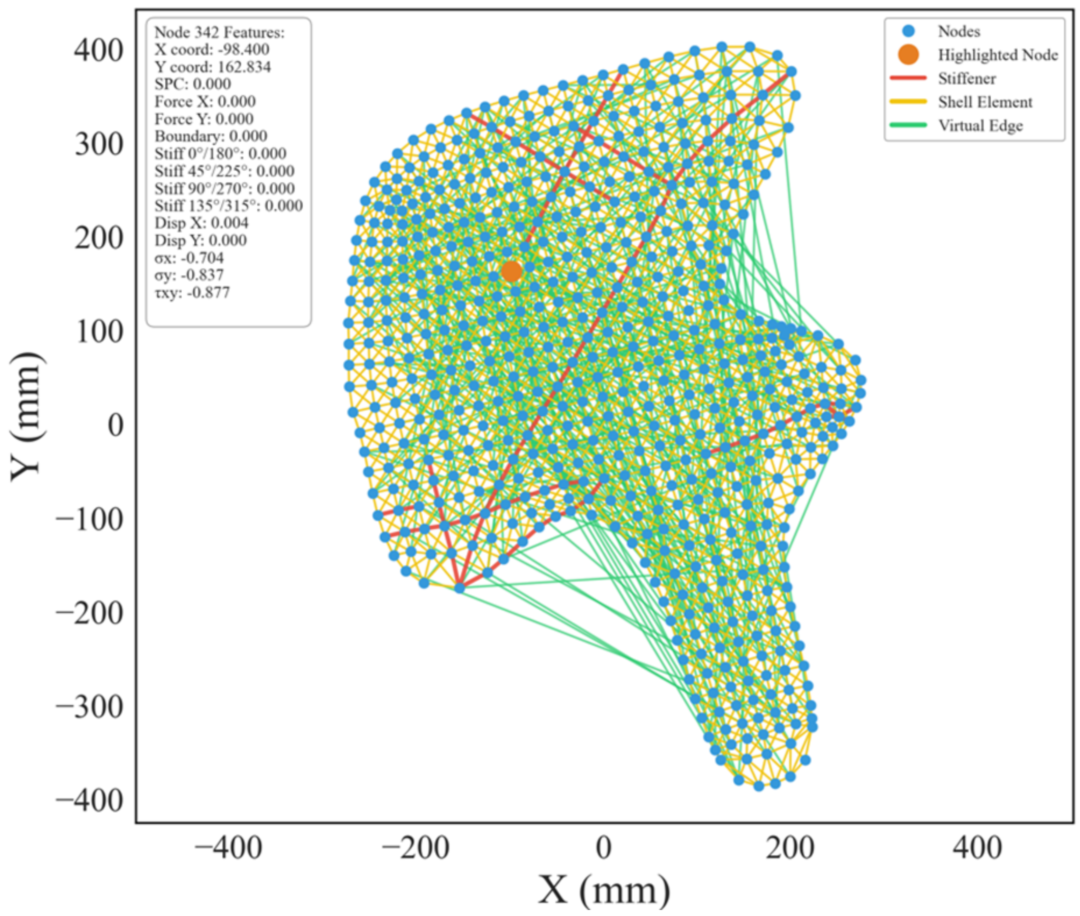


Figure 4.4: Random virtual edge generation example. Green lines show virtual edges

4.3.3 Coordinate Frame Normalization

One of the key challenges in analyzing structural shapes is ensuring consistent orientation and positioning across different geometries. To address this, we implement a robust coordinate frame normalization procedure using Principal Component Analysis (PCA). This approach ensures that all structures are aligned in a consistent manner, regardless of their original orientation, which is crucial for the neural network to learn meaningful geometric patterns and structural behaviors.

The normalization process begins by computing the covariance matrix of the node coordinates. For each structure with N nodes, we first calculate the centroid $\boldsymbol{\mu}$ and then construct the covariance matrix:

$$C = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T \tag{4.16}$$

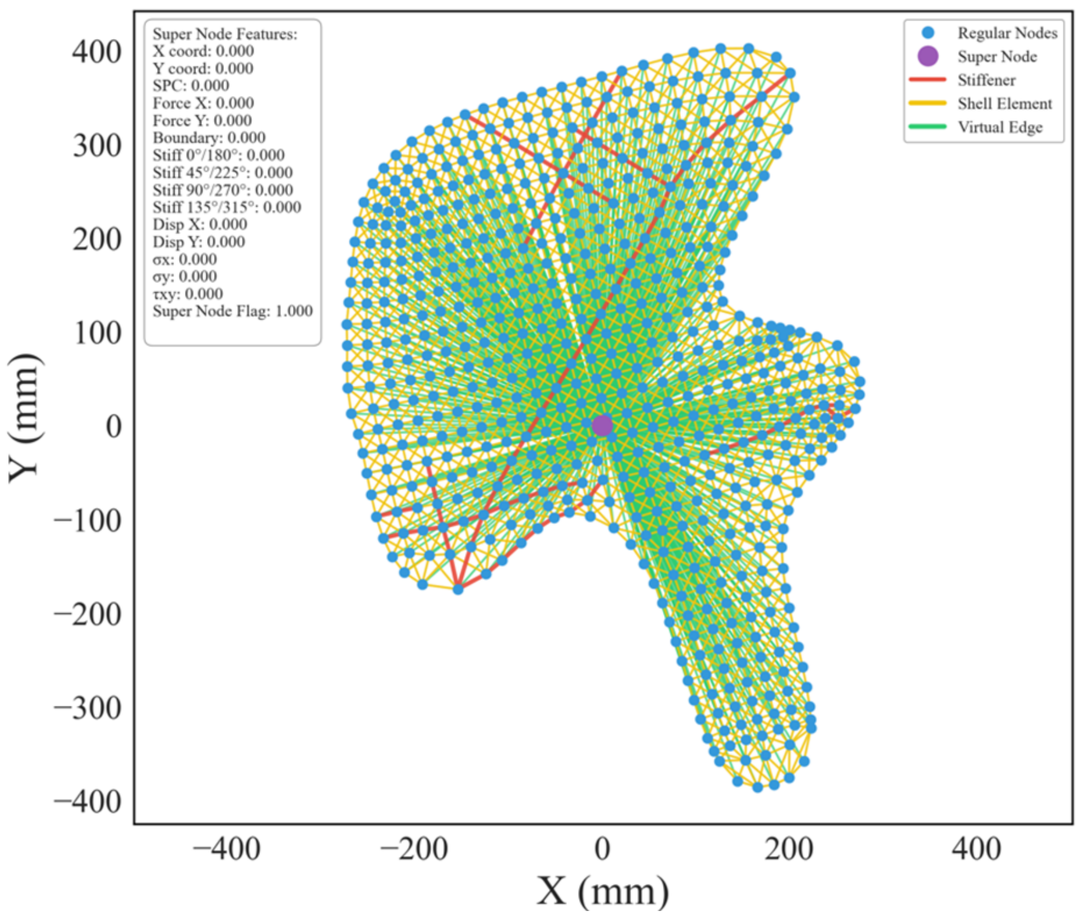


Figure 4.5: Super node generation example. Green lines show virtual edges. Purple dot represents super node. Super node features (all zero except for the super node flag) are given in the left-hand side box

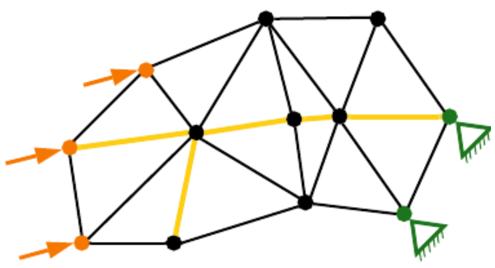
where $\mathbf{x}_i$ represents the coordinates of the i-th node. This covariance matrix captures the primary directions of variation in the structure's geometry. We then perform eigendecomposition of this matrix:

$$C\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{4.17}$$

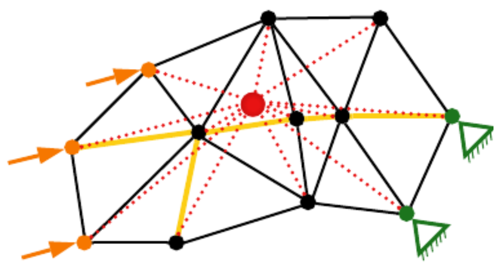
The eigenvectors $\mathbf{v}_i$ represent the principal axes of the structure, with the corresponding eigenvalues λ_i indicating the variance along each axis. To ensure consistent orientation across similar shapes, we analyze the third moment of the data projection along the primary axis:

$$m_3 = \sum_{i=1}^{N} (\mathbf{x}_i^T \mathbf{v}_1)^3 \tag{4.18}$$

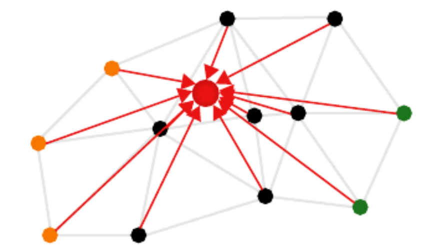
The sign of this moment determines whether the principal axis needs to be flipped to



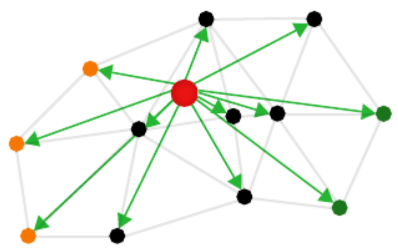
(a) Base graph



(b) Adding super node with virtual edges connected to all regular nodes



(c) Message aggregation to super node



(d) Message distribution to regular nodes

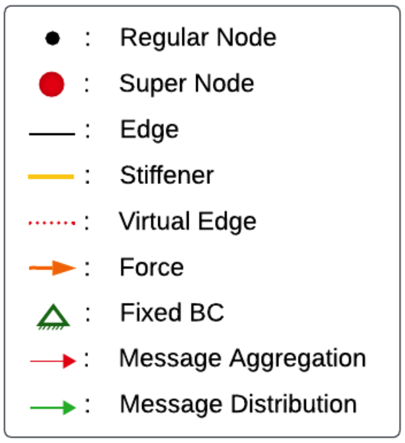


Figure 4.6: Super node architecture and information flow

maintain consistent orientation:

$$\mathbf{v}_1 = -\mathbf{v}_1 \text{ if } m_3 < 0 \tag{4.19}$$

After establishing the transformation axes, we apply a series of transformations to all relevant quantities. First, node coordinates are centered and rotated:

$$\mathbf{x}_{transformed} = R(\mathbf{x} - \boldsymbol{\mu}) \tag{4.20}$$

where R is the rotation matrix formed by the aligned eigenvectors. Vector quantities such as forces and displacements undergo similar transformation:

$$\mathbf{v}_{transformed} = R\mathbf{v} \tag{4.21}$$

Stress tensors require special handling due to their tensorial nature. We transform them using Mohr's transformation equations:

$$\begin{aligned}
\sigma'_x &= \frac{\sigma_x + \sigma_y}{2} + \frac{\sigma_x - \sigma_y}{2}\cos(2\theta) + \tau_{xy}\sin(2\theta) \\
\sigma'_y &= \frac{\sigma_x + \sigma_y}{2} - \frac{\sigma_x - \sigma_y}{2}\cos(2\theta) - \tau_{xy}\sin(2\theta) \\
\tau'_{xy} &= -\frac{\sigma_x - \sigma_y}{2}\sin(2\theta) + \tau_{xy}\cos(2\theta)
\end{aligned} \tag{4.22}$$

where θ represents the rotation angle derived from our PCA transformation. A crucial aspect of our implementation is the handling of axis flipping. When PCA indicates that coordinate axes should be flipped, we need to adjust the shear stress components accordingly. Under axis flipping:

- If x-axis is flipped: σ_x remains unchanged, τ_{xy} changes sign
- If y-axis is flipped: σ_y remains unchanged, τ_{xy} changes sign
- If both axes are flipped: all components remain unchanged

This comprehensive transformation approach ensures that our Graph Neural Network receives consistently oriented structural information, which is crucial for learning meaningful patterns and making accurate predictions. The transformation information is stored for each structure, allowing us to transform the network's predictions back to the original coordinate system when necessary.

The effectiveness of this normalization approach is proved in previous works [11], where they observe consistent prediction accuracy regardless of the initial orientation of the input structures and need for less data to train the network. This data size reduction. This invariance to rotation and translation is particularly important for our buckling analysis predictions, as it ensures that the critical loads are correctly identified regardless of the structure's original orientation and reduces data size.

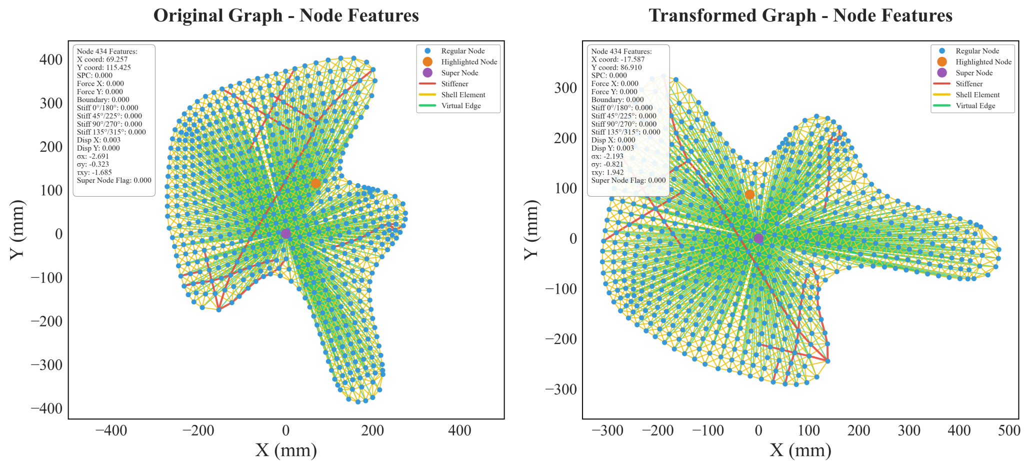


Figure 4.7: Coordinate frame normalization process and stress transformation

4.4 Model Architecture

Our neural network architecture is designed to effectively capture both local and global structural behaviors through a sophisticated message passing framework. The architecture consists of several key components: encoders that transform raw structural features into learnable representations, message passing layers that enable information flow across the structure and specialized pooling mechanisms for different prediction tasks. This design allows the network to learn complex relationships between geometric features, loading conditions and structural responses. Arctitecture layout is demonstrated in Figure 4.8

4.4.1 Node and Edge Encoders

The encoder networks serve as the foundation of our architecture, transforming raw input features into higher-dimensional latent representations that capture meaningful structural characteristics. We implement different encoder architectures based on the target hidden dimension, optimizing the network's capacity to handle varying levels of feature complexity. For node features, when working with moderate-sized hidden dimensions ($h \leq 128$), we employ a streamlined single-hidden-layer Multi-Layer

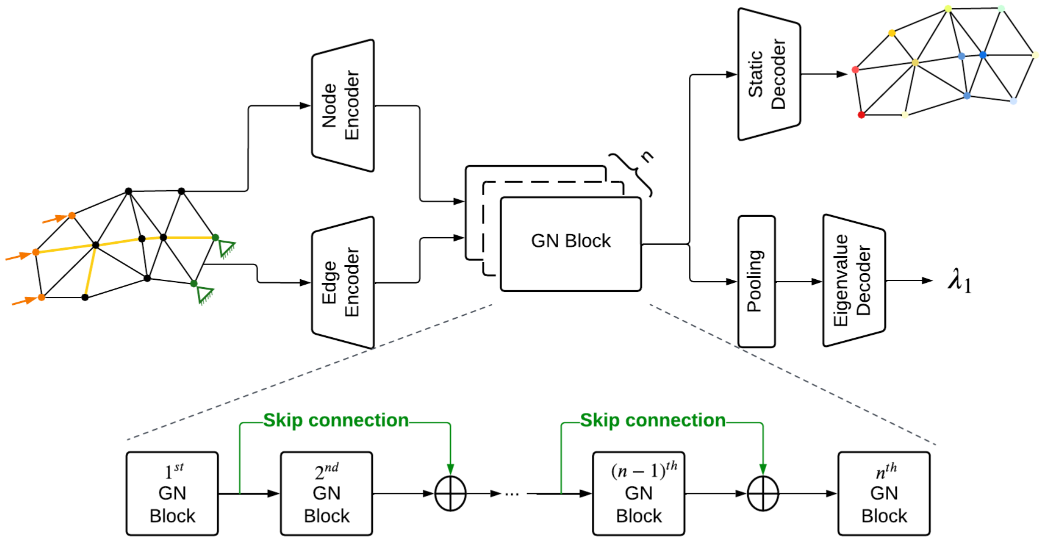


Figure 4.8: Model architecture layout

Perceptron (MLP):

$$\phi_{node}(\mathbf{x}) = \mathrm{MLP}_{64,h}(\mathbf{x}) \tag{4.23}$$

$$\mathrm{MLP}_{64,\mathrm{h}} = \begin{cases} \mathrm{Linear}(16 \to 64) \\ \mathrm{ReLU}() \\ \mathrm{Linear}(64 \to h) \end{cases} \tag{4.24}$$

Here h denotes arbitrarily selected hidden dimension number. For larger hidden dimensions ($h \geq 256$), we utilize a deeper two-hidden-layer architecture to enable more sophisticated feature transformations:

$$\phi_{node}(\mathbf{x}) = \mathrm{MLP}_{64,128,h}(\mathbf{x}) \tag{4.25}$$

The edge encoder follows a similar pattern, adapting its architecture based on the hidden dimension. For moderate dimensions:

$$\phi_{edge}(\mathbf{e}) = \mathrm{MLP}_{64,h}(\mathbf{e}) \tag{4.26}$$

And for larger dimensions:

$$\phi_{edge}(\mathbf{e}) = \mathrm{MLP}_{64,128,h}(\mathbf{e}) \tag{4.27}$$

We used MLPs in these encoders because MLPs with ReLU activation functions are universal approximators, capable of approximating any continuous function on compact subsets of R^n with arbitrary precision given sufficient hidden units. While the original universal approximation theorem focused on bounded, continuous activation functions [56], subsequent research has shown that ReLU networks also possess this universal approximation property, often with advantages in training dynamics and gradient propagation [57, 58]. These encoders are designed to maintain a balance between expressive power and computational efficiency, with the architecture depth scaling appropriately with the complexity of the feature space.

4.4.2 GraphSAGE Layer

At the heart of our architecture lies the GraphSAGE layer, which implements a sophisticated message passing mechanism through neighbor aggregation. We utilize an additive aggregation variant, which has shown superior performance in our structural analysis tasks. The core operation of each GraphSAGE layer is defined as given in Equation 4.28.

$$\mathbf{x}_v^{(k)} = \sigma\left(\mathbf{W}^k \cdot \text{AGG}(\mathbf{x}_u^{(k-1)}), \forall u \in \mathcal{N}(v)\right) \tag{4.28}$$

where $\mathbf{h}_v^{(k)}$ represents the hidden state of node v at layer k, $\mathbf{W}^k$ is a learnable weight matrix and $\mathcal{N}(v)$ denotes the neighborhood of node v. The additive aggregation function that we used in this work is specifically defined as in Equation 4.29.

$$\text{AGG}(\mathbf{x}_u) = \sum_{u \in \mathcal{N}(v)} \mathbf{x}_u \tag{4.29}$$

To facilitate effective gradient flow and enable learning of both local and global structural features, we implement skip connections between consecutive layers.

$$\mathbf{x}_v^{(k)} = \mathbf{x}_v^{(k)} + \mathbf{x}_v^{(k-1)} \text{ for } k > 1 \tag{4.30}$$

The choice of additive aggregation, rather than mean or max pooling, was motivated by its ability to better preserve magnitude information in the message passing process, which is crucial for accurate prediction of structural responses. The skip connections, connecting every other layer, help maintain access to both local and global structural information throughout the network's depth, preventing the over-smoothing problem common in deep graph neural networks.

4.4.3 CustomGNN Layer

Our custom Graph Neural Network layer, inspired by MeshGraphNets [9] and further refined based on EA-GNN architecture [11], implements a sophisticated message passing mechanism that explicitly incorporates edge attributes. This design is particularly effective for structural analysis as it can directly process geometric relationships encoded in edge features, such as element orientations and connectivity information. This layer is referred as CustomGNN throughout the thesis.

4.4.3.1 Message Passing Operations

The layer operates through a sequence of three main operations that work together to update both edge and node representations.

Edge Update Operation: The first step involves updating edge features by considering both the connected nodes and the edge's current features.

$$\mathbf{e}'_{ij} = \chi(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ij}) \tag{4.31}$$

Here, $\mathbf{x}_i$ and $\mathbf{x}_j$ represent the features of nodes connected by edge (i, j) and $\mathbf{e}_{ij}$ represents the edge's current features. The edge update function χ is implemented as a single layer MLP.

$$\chi = \text{MLP}_{h,h}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ij}) \tag{4.32}$$

This operation allows edges to adapt their representations based on the states of their incident nodes.

Message Computation and Aggregation: After updating edge features, the layer computes messages between connected nodes.

$$\mathbf{m}_{ij} = \phi(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}'_{ij}) \tag{4.33}$$

These messages are then aggregated for each node using a mean operation.

$$\mathbf{m}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \tag{4.34}$$

The mean aggregation helps normalize the incoming information, preventing scaling issues in nodes with different numbers of neighbors.

Node Update Operation: Finally, nodes update their features using two steps.

$$\begin{aligned} \mathbf{x}_i &= \gamma(\mathbf{x}_i, \mathbf{m}_i) \\ \mathbf{x}'_i &= \mathbf{x}_i + \beta(\mathbf{x}_i) \end{aligned} \tag{4.35}$$

The functions involved in these operations are implemented as MLPs with single hidden layer.

$$\phi = \text{MLP}_{h,h}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}'_{ij}) \tag{4.36}$$

$$\gamma = \text{MLP}_{h,h}(\mathbf{x}_i, \mathbf{m}_i) \tag{4.37}$$

$$\beta = \text{MLP}_{h,h}(\mathbf{x}_i) \tag{4.38}$$

After node update is done, message propagation in the layer is completed.

4.4.3.2 Layer Architecture

The layer's MLPs are carefully designed to maintain consistent feature dimensions while enabling rich feature transformations.

Edge MLP (χ) processes concatenated node and edge features:

$$\text{MLP}_{\text{edge}} = \begin{cases} \text{Linear}(3h \to h) \\ \text{ReLU}() \\ \text{Linear}(h \to h) \end{cases} \tag{4.39}$$

Message MLP (ϕ) handles message aggregation:

$$\text{MLP}_{\text{node}} = \begin{cases} \text{Linear}(2h \to h) \\ \text{ReLU}() \\ \text{Linear}(h \to h) \end{cases} \tag{4.40}$$

Node MLP (γ) handles concatenated node features and message information:

$$\text{MLP}_{\text{node}} = \begin{cases} \text{Linear}(2h \to h) \\ \text{ReLU}() \\ \text{Linear}(h \to h) \end{cases} \tag{4.41}$$

Residual MLP (β) maintains feature dimensionality:

$$\text{MLP}_{\text{residual}} = \begin{cases} \text{Linear}(h \to h) \\ \text{ReLU}() \\ \text{Linear}(h \to h) \end{cases} \tag{4.42}$$

Skip connections are implemented for both nodes and edges to facilitate gradient flow and preserve information across layers.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{x}_{k+1} \text{ for nodes} \tag{4.43}$$

$$\mathbf{e}_{k+1} = \mathbf{e}_k + \mathbf{e}_{k+1} \text{ for edges} \tag{4.44}$$

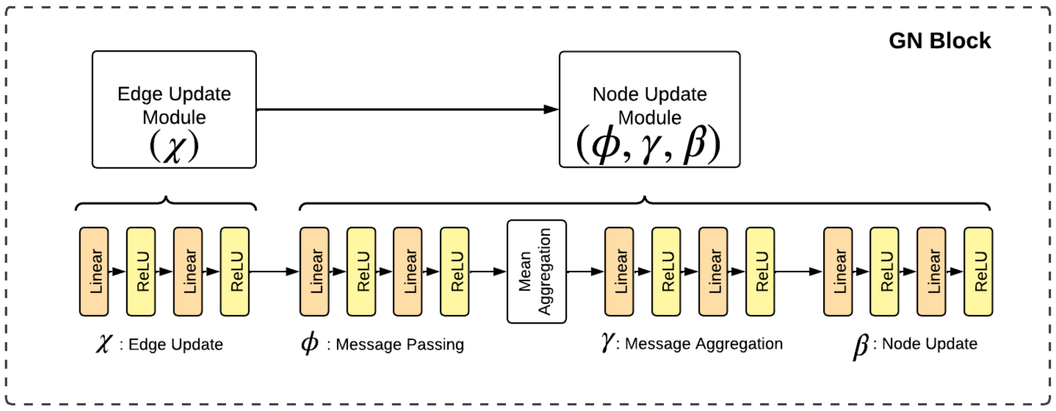
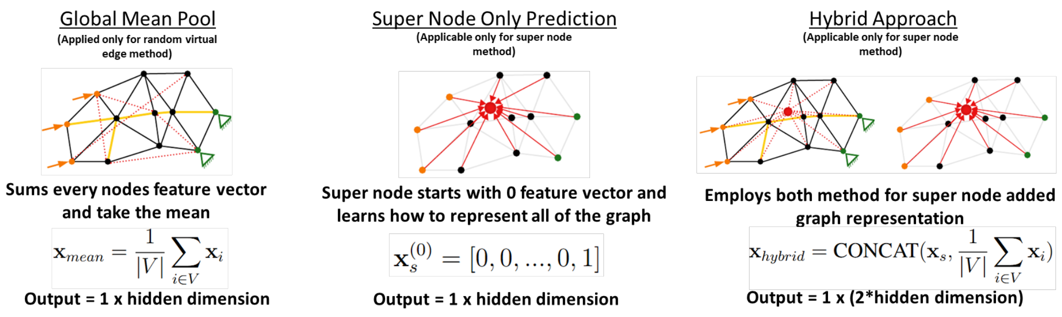

GN Block
Edge Update Module (χ)
Node Update Module (ϕ, γ, β)
Linear
ReLU
Mean Aggregation
χ : Edge Update
ϕ : Message Passing
γ: Message Aggregation
β: Node Update

Figure 4.9: CustomGNN layer architecture

This is the second architecture that we use in our thesis, where architectural design is shown in Figure 4.9. This architecture's explicit edge processing enables better representation of structural connectivity and geometric relationships. GraphSAGE can not utilize edge features whereas this GNN block can utilize this explicit knowledge.

4.4.4 Pooling Strategies

While there is no need for pooling layers for node level prediction implementations, for graph level prediction task like predicting buckling eigenvalues, a pooling layer is needed before the decoder to aggregate updated node information from the whole graph. Pooling mechanisms play a crucial role in our architecture for predicting buckling eigenvalues. Through extensive experimentation, we developed and evaluated several pooling strategies, each offering distinct advantages for buckling eigenvalue prediction tasks. Our investigation led to three main approaches: super node only pooling, hybrid pooling combining super node with mean aggregation and traditional mean pooling.



Global Mean Pool
(Applied only for random virtual edge method)
Sums every nodes feature vector and take the mean
$\mathbf{x}_{mean} = \frac{1}{|V|}\sum_{i \in V} \mathbf{x}_i$
Output = 1 x hidden dimension
Super Node Only Prediction
(Applicable only for super node method)
Super node starts with 0 feature vector and learns how to represent all of the graph
$\mathbf{x}_s^{(0)} = [0, 0, ..., 0, 1]$
Output = 1 x hidden dimension
Hybrid Approach
(Applicable only for super node method)
Employs both method for super node added graph representation
$\mathbf{x}_{hybrid} = \text{CONCAT}(\mathbf{x}_s, \frac{1}{|V|}\sum_{i \in V} \mathbf{x}_i)$
Output = 1 x (2*hidden dimension)

Figure 4.10: Pooling strategies using different approaches

As it will be understand in the upcoming chapter, the super node only pooling strategy emerged as the most effective approach for eigenvalue prediction. This success can be attributed to the super node's unique ability to learn and represent global structural characteristics. During message passing iterations, the super node accumulates information from all nodes in the structure, effectively creating a comprehensive representation of the structure's overall stiffness and load distribution patterns.

The mathematical representation of this process is node update processes of each framework. Equation 4.28 shows node feature update for GraphSAGE framework and Equation 4.35 shows node feature update for custom GNN framework. After node features updated and finalized, only final super node features are used to predict buckling eigenvalue.

A key insight from our research was the importance of the super node's initial state. By initializing the super node with zero features except for its type indicator, we allow it to learn purely from the structure's characteristics without any predetermined biases.

$$\mathbf{x}_s^{(0)} = [0, 0, ..., 0, 1] \tag{4.45}$$

The final value (1) indicates the super node type. We also implemented a hybrid pooling mechanism that combines super node features with mean-pooled features from regular nodes.

$$\mathbf{x}_{hybrid} = \text{CONCAT}(\mathbf{x}_s, \frac{1}{|V|}\sum_{i \in V} \mathbf{x}_i) \tag{4.46}$$

While this approach provides redundancy in global information gathering, our experimental results showed that it did not significantly improve prediction accuracy over the simpler super node only approach. This pooling operation also requires double sized hidden dimension for decoder because of concatenation of super node features and global mean pooling values.

The traditional mean pooling strategy is selected to be only pooling layer option for random virtual edge augmented models. These models do not contain any super node instead they include random virtual edges. Because of that we cannot use previous pooling layer options.

$$\mathbf{x}_{mean} = \frac{1}{|V|}\sum_{i \in V} \mathbf{x}_i \tag{4.47}$$

These three pooling layers are used throughout this study for buckling eigenvalue predictions.

4.4.5 Decoder Architecture

The decoder architecture in our framework serves as the final transformation stage, converting learned graph representations into prediction outputs. The design of our decoder networks reflects the fundamental difference between predicting global scalar properties (like buckling eigenvalues) and local field quantities (like displacements). This distinction led us to develop specialized decoder architectures for each prediction task.

For buckling eigenvalue prediction, our decoder processes the highly concentrated information from the super node through a carefully designed multi-layer network. The architecture's depth and width were determined through extensive experimentation, leading to different optimal configurations based on the model's hidden dimension size. For models with hidden dimensions up to 128, we employ a single hidden layer compact architecture.

$$\phi_{bd}(\mathbf{x}) = \text{MLP}_{h/2,1}(\mathbf{x}) \tag{4.48}$$

$$\phi_{bd} = \begin{cases} \text{Linear}(h \to 64) \\ \text{ReLU}() \\ \text{Linear}(64 \to 1) \end{cases} \tag{4.49}$$

This compact design proved sufficient for processing the concentrated information from the super node. However, for larger models with hidden dimensions of 256 or greater, we try to ensure gradually decrease by stepping down the feature dimension in stages smoothly (e.g., 256 $\to$ 128 $\to$ 64 $\to$ 1), we allow the network to gradually distill the complex structural information into a single predictive value while maintaining important feature relationships.

$$\phi_{bd}(\mathbf{x}) = \text{MLP}_{128,64,1}(\mathbf{x}) \tag{4.50}$$

$$\phi_{bd} = \begin{cases} \text{Linear}(h \to 128) \\ \text{ReLU}() \\ \text{Linear}(128 \to 64) \\ \text{ReLU}() \\ \text{Linear}(64 \to 1) \end{cases} \tag{4.51}$$

For displacement field prediction, the decoder takes on a different role. Instead of concentrating information, it must transform node-wise features into local displacement predictions while maintaining spatial relationships. Our implementation for this task utilizes a shared MLP across all nodes.

$$\mathbf{d}_i = \phi_{fd}(\mathbf{x}_i) \quad \forall i \in V \tag{4.52}$$

where $\mathbf{d}_i$ represents the predicted displacement components at node i. The decoder architecture for this task maintains the feature dimension longer in its processing pipeline. Because dimension of the displacement field is 2, namely u_x and u_y, output of the decoder is 2 representing [u_x, u_y].

$$\phi_{fd}(\mathbf{x}) = \text{MLP}_{64,ds}(\mathbf{x}) \tag{4.53}$$

$$\phi_{fd} = \begin{cases} \text{Linear}(h \to 64) \\ \text{ReLU}() \\ \text{Linear}(64 \to 2) \end{cases} \tag{4.54}$$

One of the key point is the importance of the activation function placement in the decoder. While ReLU activations were crucial for the intermediate layers adding nonlinearity to the model, maintaining linearity in the final layer is essential for accurate prediction of both positive and negative values in displacement components. This design choice was particularly important for capturing the full range of structural responses.

4.5 Loss Function Design

The design of appropriate loss functions proved to be crucial for achieving accurate predictions in structural analysis tasks. Our research revealed that standard loss functions sometimes fell short in capturing the unique characteristics of structural behavior, leading us to develop specialized loss functions for different prediction tasks.

4.5.1 Buckling Analysis Loss Functions

In buckling analysis, the accurate prediction of eigenvalues across different orders of magnitude presented a particular challenge. Traditional mean squared error (MSE) loss tended to prioritize larger eigenvalues while paying insufficient attention to smaller, yet equally important, values. To address this, we utilized relative error loss function that provides balanced training across the full range of eigenvalues.

$$\mathcal{L}_{rel} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{4.55}$$

Ground truth buckling eigenvalues for a graph is represented by y and model's predicted buckling eigenvalue for the same graph is represented by ŷ.

4.5.2 Static Analysis Loss Functions

For static analysis, we developed a specialized graph-wise Mean Absolute Error (MAE) loss function that accounts for the hierarchical nature of our graph-based predictions. In this context, y represents the ground truth displacement values at each node, while ŷ represents the model's predicted displacement values at those same nodes. Unlike traditional MAE implementations that treat all nodes across the batch uniformly, our approach preserves the graph structure by computing the error separately for each graph before averaging, ensuring that graphs with different numbers of nodes are weighted equally in the final loss calculation. The loss function is formulated as follows:

$$\mathcal{L}_{static} = \frac{1}{G}\sum_{g=1}^{G}\left(\frac{1}{N_g}\sum_{i=1}^{N_g}|y_i^g - \hat{y}_i^g|\right) \tag{4.56}$$

where G represents the total number of graphs in the batch, N_g is the number of nodes in graph g, y_i^g is the true displacement at node i of graph g and $\hat{y}_i^g$ is the predicted displacement at the same node. This formulation first computes the mean absolute error for each graph independently, then averages these errors across all graphs in the batch. This approach ensures that each graph contributes equally to the loss regardless of its size, preventing larger graphs from dominating the training signal.

4.5.3 Loss Function Scheduler

To ensure efficient training, we implemented cosine annealing scheduler [59]. The scheduler operates with specified base period, gradually reducing the learning rate according to the cosine function.

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + \cos(\frac{t\pi}{T})) \tag{4.57}$$

where η_t is the learning rate at epoch t, η_{min} and η_{max} are the minimum and maximum learning rates respectively and T is the total number of epochs.

4.6 Implementation Details

Our software implementation builds upon PyTorch [60] and PyTorch Geometric (PyG) [61], chosen for their combination of flexibility and performance. PyTorch's dynamic computational graphs proved particularly valuable during development, allowing rapid prototyping and easier debugging of complex graph operations. The integration with PyG provided essential utilities for handling graph-structured data, significantly reducing the complexity of implementing message-passing operations.

As for training infrastructure, our setup employed TESLA V100-SXM2 and TESLA P100 GPUs with 16GB VRAM. Access to these resources was granted by TRUBA. The batch size of 16 was chosen based on GPU memory constraints

CHAPTER 5

MODEL TRAINING AND RESULTS

5.1 Introduction

This chapter presents a comprehensive analysis of our Graph Neural Network framework's training process and performance, with a particular focus on buckling eigenvalue prediction. Our investigation follows a systematic two-phase approach: first optimizing the architecture and hyperparameters using a base dataset of non-stiffened structures, then applying these optimized configurations to the more complex domain of stiffened plates. Our implementation leverages architectural design detailed in Chapter 4, where technical background is given in every detail.

This methodological progression from simpler to more complex structural configurations serves multiple purposes. First, it enables thorough validation of our architectural choices and training strategies in a more controlled setting. The base dataset, comprising 40,000 selected samples from an initial pool of 200,000 analyses, provides a foundation for hyperparameter optimization and architectural comparisons. This approach allows us to establish clear baseline performance metrics before tackling the additional complexities introduced by stiffener elements. The subsequent application to our expanded dataset of 80,000 stiffened structures, selected from over 700,000 analyses, demonstrates the framework's scalability and adaptability to more complex structural configurations.

While our framework was designed to handle multiple prediction tasks, including displacement field prediction, our investigation revealed particularly promising results in buckling eigenvalue prediction. The challenges encountered in the prediction of the displacement field, where our implementation of the EA-GNN architecture [11] and the GraphSAGE architecture [20] showed limited success compared to published results, led us to investigate the underlying reasons.

Throughout this chapter, we present a detailed analysis of our training methodology, results and insights gained from both phases of our investigation. We examine model performance across selected geometric configurations and loading conditions providing an understanding of our framework's capabilities and limitations. Special attention is paid to generalization ability, demonstrated through performance on modified test cases including scaled geometries.

The structure of this chapter progresses from training framework implementation to detailed results analysis. We begin by examining hyperparameter optimization process, followed by comprehensive results from both non-stiffened and stiffened datasets. The analysis concludes with a comparative analysis against traditional finite element methods, providing insights into the framework's potential impact on structural engineering practice.

5.2 Hyperparameter Optimization

The development of an effective Graph Neural Network architecture for structural analysis requires careful tuning of multiple hyperparameters that significantly influence model performance. Our optimization strategy focused on four key architectural aspects: model type selection, hidden dimension size, number of message passing layers, and pooling strategy. We conducted a grid search across these parameters, evaluating more than 220 different model configurations to identify the optimal architecture for buckling prediction.

5.2.1 Grid Search Implementation

Our grid search methodology systematically explored the following hyperparameter space:

- Model Architecture: GraphSAGE vs CustomGNN

- Hidden Dimensions: [64, 128, 256, 512]
- Number of Layers: [2, 3, 4, 6]
- Pooling Strategies: [mean, supernode, hybrid]

To evaluate performance of the models, MAPE metric is used:

$$\text{MAPE} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100 \tag{5.1}$$

Figure 5.1 shows the validation MAPE trajectories for all configurations tested during our grid search process. The dense collection of curves, each representing a unique combination of hyperparameters, demonstrates the extensive nature of our optimization effort. This visualization reveals significant variation in model performance across different configurations, with final validation MAPE values ranging from approximately 8% to over 60%.

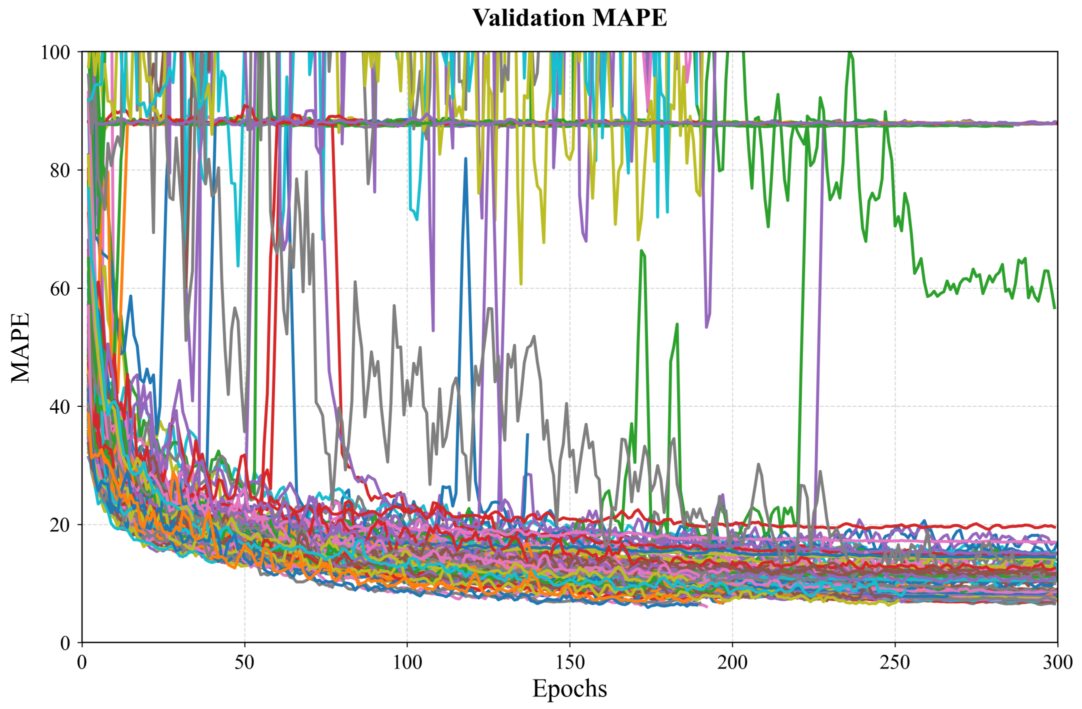


Figure 5.1: Validation MAPE trajectories for all hyperparameter configurations tested during grid search optimization

5.2.2 Model Architecture Selection

The first key decision in our optimization process was the choice between GraphSAGE and CustomGNN architectures. As shown in Figure 5.2, we compared these architectures while maintaining consistent settings for other hyperparameters (hidden dimension: 256, number of layers: 6, hybrid pooling).

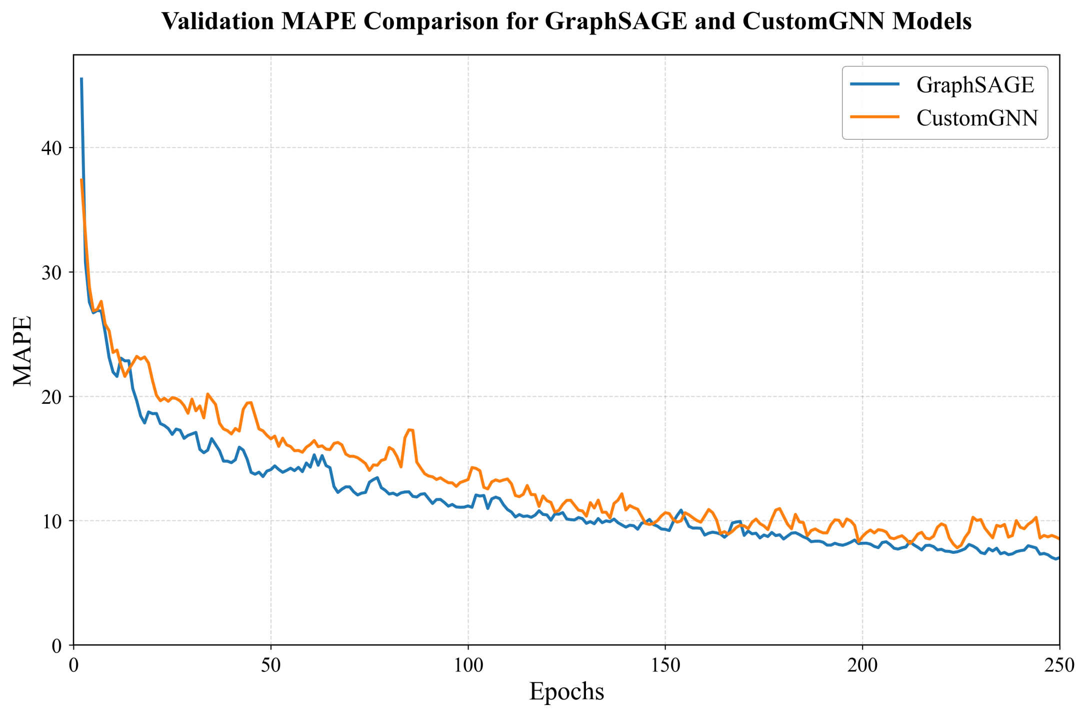


Figure 5.2: Validation MAPE comparison between GraphSAGE and CustomGNN architectures

GraphSAGE demonstrated consistently superior performance, achieving lower validation MAPE values throughout the training process. By epoch 250, GraphSAGE achieved a validation MAPE of approximately 7.5% compared to CustomGNN's 9%. This performance advantage can be attributed to GraphSAGE's efficient neighborhood aggregation mechanism and its ability to learn node embeddings that effectively capture both local and global structural characteristics.

5.2.3 Hidden Dimension Analysis

We investigated the impact of hidden dimension size on model performance, testing four different dimensions: 64, 128, 256, and 512. Figure 5.3 presents the validation MAPE curves for each dimension setting.

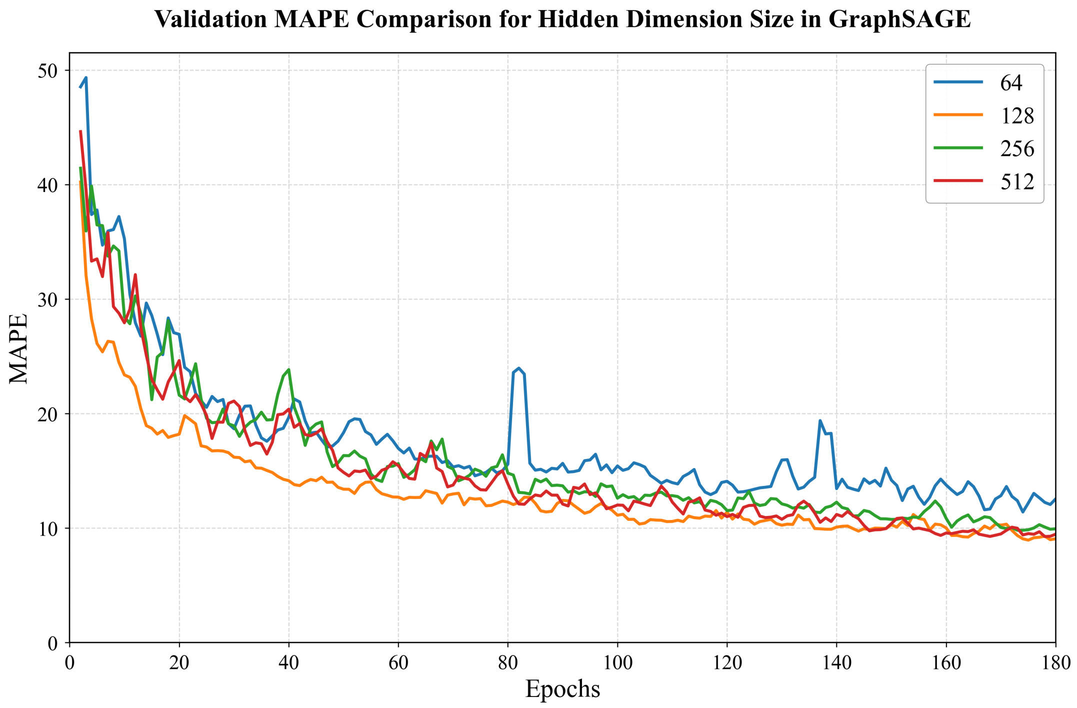


Figure 5.3: Validation MAPE comparison across different hidden dimension sizes

The results reveal that while a hidden dimension of 64 consistently underperformed, the differences between larger dimensions (128, 256 and 512) were less pronounced. However, the 512-dimensional model showed marginally better performance, particularly in the later epochs. This suggests that while increased representational capacity benefits model performance, the returns diminish beyond a certain point. The 512-dimensional model's superior performance justifies its selection despite its higher computational cost.

5.2.4 Layer Depth Analysis

The number of message passing layers significantly influences the model's ability to capture multi-scale structural behaviors. We evaluated configurations with 2, 3, 4, and 6 layers, as shown in Figure 5.4.

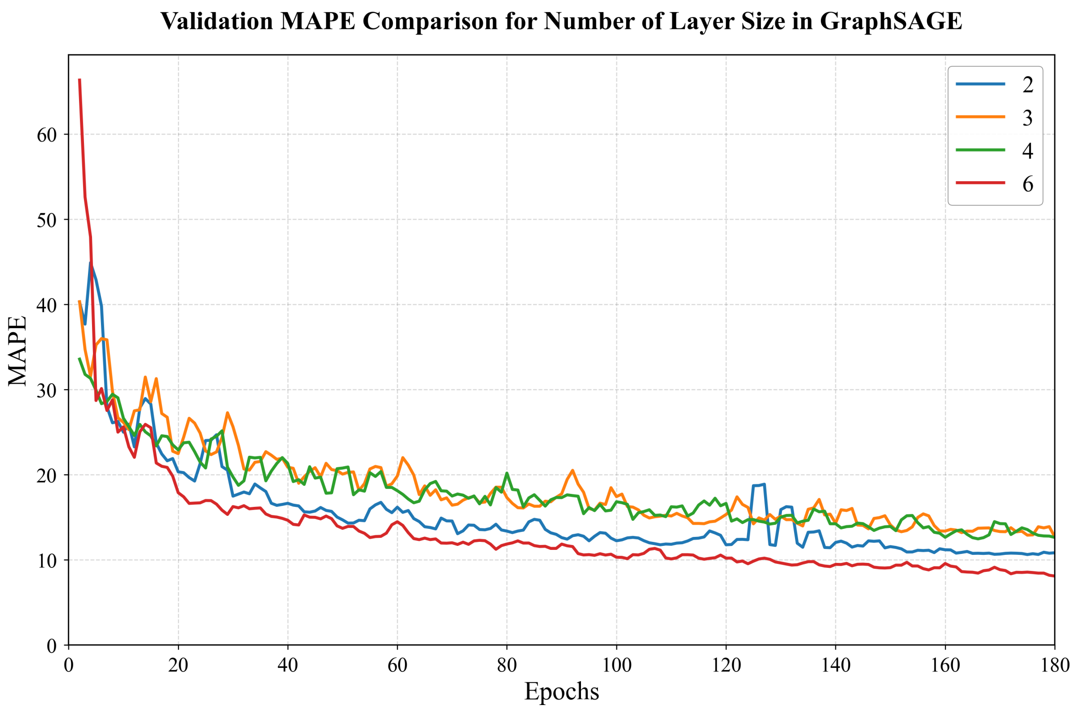


Figure 5.4: Validation MAPE comparison for different numbers of message passing layers

Interestingly, both the shallowest (2 layers) and deepest (6 layers) configurations showed strong performance. The 6-layer model ultimately achieved the lowest validation MAPE, suggesting that deeper architectures can better capture the complex relationships in structural analysis. However, the strong performance of the 2-layer model indicates that even relatively shallow networks can learn meaningful representations for buckling prediction, particularly in non-stiffened structures where the geometric complexity is lower.

5.2.5 Pooling Strategy Evaluation

The final aspect of our optimization focused on pooling strategies: mean pooling, supernode pooling, and hybrid pooling. Figure 5.5 presents the comparative performance of these approaches.

Both hybrid and supernode pooling outperformed mean pooling, with hybrid pooling

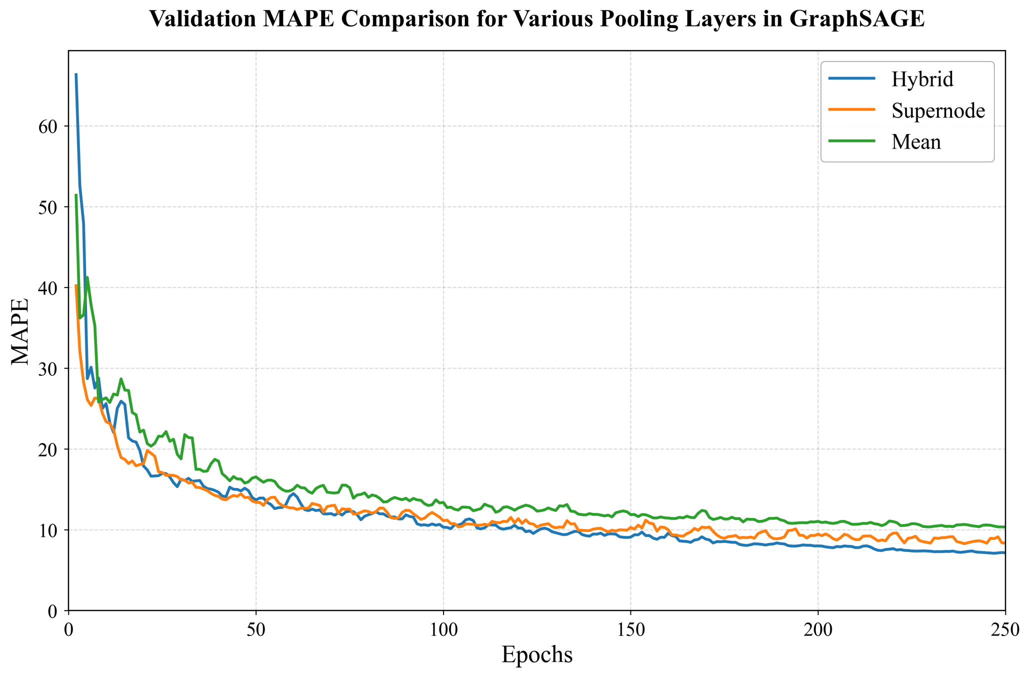


Figure 5.5: Validation MAPE comparison for different pooling strategies

showing better results. The superior performance of supernode and hybrid approaches demonstrates the importance of maintaining global structural information when predicting buckling behavior. Mean pooling's relatively poor performance suggests that simple averaging of node features loses crucial information about the structural system's global characteristics.

5.2.6 Optimal Configuration and Training Results for Base Dataset

Based on our comprehensive hyperparameter optimization study, we identified the following optimal configuration.

- Architecture: GraphSAGE with additive aggregation
- Hidden Dimension: 512
- Number of Layers: 6
- Pooling Strategy: Hybrid
- Dropout Rate: 0.1

This configuration demonstrated the best overall performance. This configuration will be used to train both non-stiffened and stiffened dataset.

Using this configuration we achieved a final validation MAPE of 5.5% on the base dataset. The success of this architecture can be attributed to its ability to effectively balance local feature extraction through deep message passing with global information preservation through hybrid pooling.

The relatively small gap between training and validation MAPE indicates that the model successfully avoids overfitting while capturing the underlying physical relationships governing buckling behavior. This configuration was subsequently used as the foundation for our investigations with stiffened structures, which we discuss in the following section.

5.3 Model Training and Performance Analysis

The evaluation of our Graph Neural Network architecture spanned two distinct datasets: a base dataset consisting of non-stiffened structures and an enhanced dataset incorporating stiffened structures. This section presents a analysis of model performance across both scenarios.

5.3.1 Base Dataset Training Results

Training on the base dataset of non-stiffened structures served as our initial validation of the model architecture. Using the optimal configuration identified through hyperparameter optimization, we achieved remarkable prediction accuracy for buckling eigenvalues. Figure 5.6 illustrates the training progression over 450 epochs.

The model demonstrated strong learning characteristics on the base dataset, achieving a final validation MAPE of 5.5%. Several key observations can be made from the training curves. First, rapid initial convergence during the first 50 epochs indicates efficient learning of fundamental structural patterns. Minimal gap between training and validation curves suggests robust generalization. Model performance is stable, minimal oscillation in later epochs.

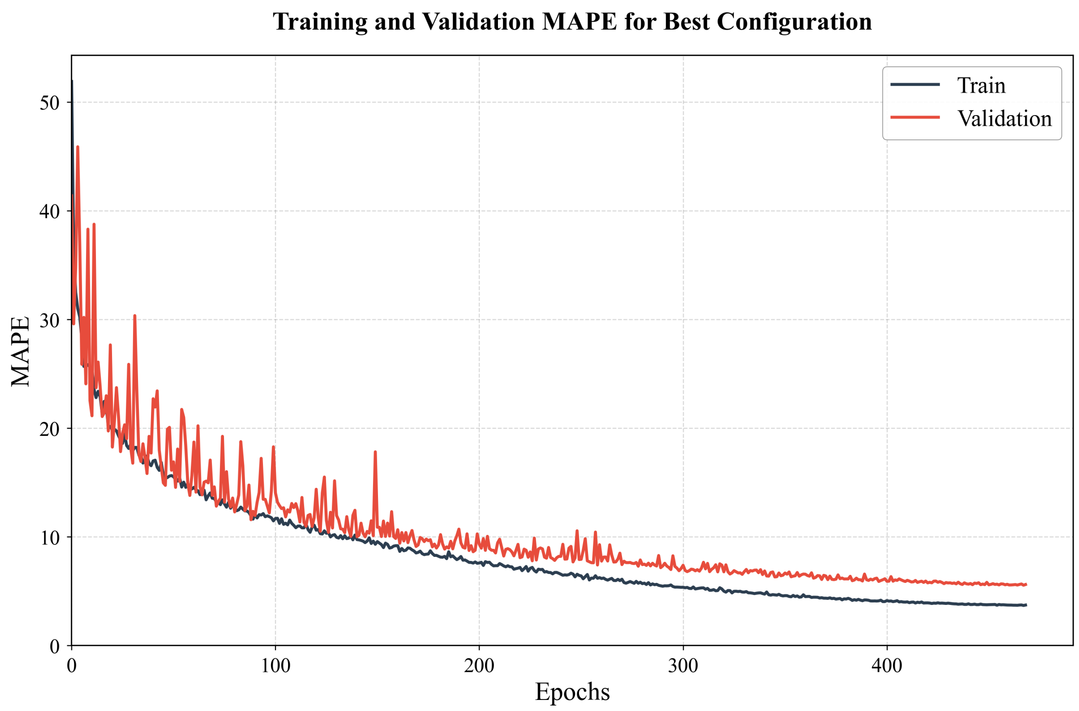


Figure 5.6: Training and validation MAPE curves for the base dataset showing consistent convergence and minimal overfitting

The low validation MAPE of 5.5% represents excellent prediction accuracy for buckling eigenvalues, particularly considering the diverse range of geometries and loading conditions in the dataset. This performance level indicates that the model successfully learned to capture the complex relationships between structural geometry, loading conditions, and buckling behavior.

5.3.2 Stiffened Dataset Performance

Following the successful validation on the base dataset, we applied the same architectural configuration to the more complex stiffened structures dataset. This represents a significant increase in problem complexity, as the addition of stiffeners introduces new geometric features and structural interactions. Figure 5.7 shows the training progression for the stiffened dataset.

The model achieved a validation MAPE of 12.5% on the stiffened dataset, with several notable characteristics in the training process:

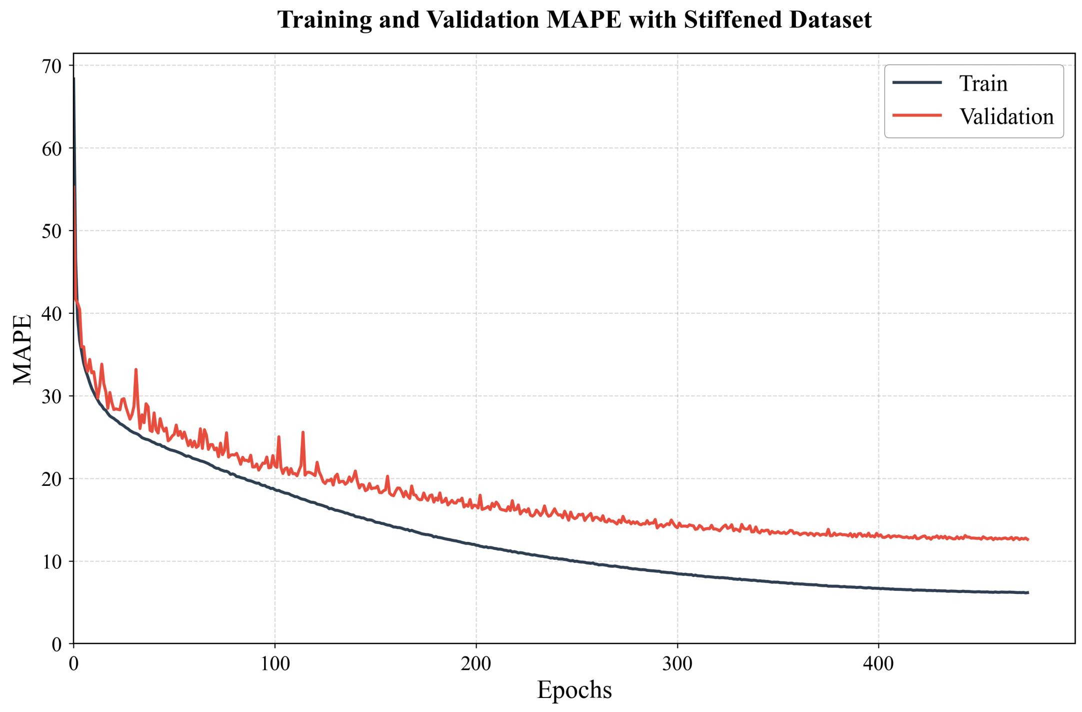


Figure 5.7: Training and validation MAPE curves for the stiffened dataset demonstrating reasonable learning despite increased structural complexity

- Higher initial MAPE values compared to the base dataset, reflecting increased problem complexity
- More pronounced oscillations in validation error, indicating more challenging optimization landscape
- Longer convergence period, suggesting more complex feature relationships requiring extended learning
- Slightly larger gap between training and validation curves, though still within acceptable bounds

The increase in validation MAPE from 5.5% to 12.5% when moving from base to stiffened structures is reasonable given the substantial increase in problem complexity. The addition of stiffeners introduces new geometric features, local-global interactions, and more complex load paths that the model must learn to process. Despite this increased complexity, the model maintains acceptable prediction accuracy, demonstrating its robustness and scalability to more challenging structural configurations.

The successful application of our architecture to both datasets validates its effectiveness as a surrogate model for buckling analysis. The consistent performance across different levels of structural complexity suggests that the fundamental architectural choices - particularly the combination of GraphSAGE layers with super node and hybrid pooling - provide a robust foundation for buckling analysis tasks.

5.4 Model Generalization and Robustness

The evaluation of our Graph Neural Network's generalization capabilities involved comprehensive testing across multiple scenarios, examining both non-stiffened and stiffened structures under various conditions. Table 5.1 presents the complete test results for our optimized GraphSAGE model with 512 hidden dimensions and 6 message passing layers.

Table 5.1: Test Results for GraphSAGE Model

Structure Type	**Test Case**	**MAPE (%)**	**Min Error (%)**	**Max Error (%)**
Non-Stiffened	Base	6.73	0.0004	78.03
	0.5× Scale	16.71	0.025	35.55
	2.0× Scale	29.97	0.006	181.63
	Multiple Loading	13.46	0.002	296.73
Stiffened	Base	17.64	0.002	214.59
	0.5× Scale	24.17	0.039	53.45
	2.0× Scale	45.44	0.009	255.39
	Multiple Loading	20.90	0.002	462.59

5.4.1 Base Performance Analysis

The model's performance on the base test set provides our primary benchmark for generalization capability. For non-stiffened structures, the achieved MAPE of 6.73% represents excellent prediction accuracy, particularly when compared to the validation performance of 5.5% during training. This small difference between validation and test performance (approximately 1.2 percentage points) indicates robust generalization to unseen geometries generated with the same parameters as the training set.

For stiffened structures, the base test MAPE of 17.64% compared to the validation MAPE of 12.5% shows a more pronounced generalization gap of about 5 percentage points. This increased gap aligns with our expectations given the additional complexity introduced by stiffener elements and their interactions with the base structure. The higher maximum error (214.59% versus 78.03% for non-stiffened) suggests that certain stiffener configurations produce more challenging prediction scenarios, particularly when combined with complex loading conditions.

5.4.2 Scale Invariance Analysis

Our scale invariance tests reveal systematic patterns in model performance across different geometric scales. For non-stiffened structures, scaling to half size (0.5×) results in a MAPE increase to 16.71%, while doubling the size (2.0×) leads to a more substantial increase to 29.97%. Notably, the maximum error for 0.5× scaling (35.55%) is lower than the base case (78.03%), suggesting that while overall accuracy decreases, prediction stability actually improves for smaller scales.

The stiffened structures exhibit more pronounced scaling effects, with MAPE values increasing to 24.17% and 45.44% for 0.5× and 2.0× scales respectively. This scale sensitivity appears to be amplified by the presence of stiffeners, possibly due to the changing relative significance of stiffener-plate interactions at different scales. The maximum errors follow a similar pattern, though interestingly, the 0.5× scale shows substantially lower maximum error (53.45%) compared to both base and 2.0× configurations. These scaling effects can be attributed to several factors. The model's training on a specific size range (700mm to 1000mm) creates an implicit scale bias despite our coordinate normalization strategy. The physical behavior of structures can change with scale, particularly regarding the relative importance of different buckling modes. The relationship between geometric features and buckling behavior may become nonlinear at scales significantly different from the training range.

5.4.3 Complex Loading Response Analysis

The introduction of multiple loading lines and boundary conditions provides insight into the model's robustness to loading complexity. For non-stiffened structures, the MAPE of 13.46% represents a reasonable degradation from base performance, considering the substantially increased complexity of the loading scenarios. However, the maximum error of 296.73% indicates potential stability issues in extreme cases.

Stiffened structures show similar sensitivity to loading complexity, with MAPE increasing to 20.90% and maximum errors reaching 462.59%. This substantial increase in maximum error suggests that the combination of stiffeners and complex loading patterns can create particularly challenging prediction scenarios, possibly due to complex load path interactions through the stiffener network.

5.4.4 Model Limitations

Analysis of error distributions across test configurations reveals important patterns that help us understand both the capabilities and limitations of our Graph Neural Network framework. The most striking pattern emerges in the relationship between structural complexity and prediction accuracy. When stiffeners are introduced to the structural system, we observe not only an increase in mean error but also a notably higher variance in predictions. This effect becomes particularly pronounced in cases where the geometric scale deviates significantly from the training dataset, reaching its peak in the 2.0× scale configurations. The interaction between stiffeners and scale appears to create a compounding effect, suggesting that our model struggles to maintain consistent accuracy when multiple complexity factors coincide.

The challenge of predicting buckling behavior becomes even more evident when examining structures under complex loading conditions. Our analysis shows that multiple loading configurations generate the broadest error distributions in our test cases, with this effect being particularly pronounced in stiffened structures. This observation suggests that the complexity of loading patterns poses a more significant challenge to our model's predictive capabilities than purely geometric variations. The model appears to have greater difficulty capturing the intricate load paths and stress distributions that arise from multiple loading points, especially when these complex loading patterns interact with stiffener elements.

Through our comprehensive testing program, we have identified clear boundaries where our model's reliability begins to diminish. A particularly clear limitation emerges in the model's handling of geometric scaling. Performance degradation becomes significant when structures deviate beyond twice their original size, effectively establishing practical application limits between half and double the original scale range. This scaling limitation likely stems from the model's training distribution and suggests that the coordinate normalization strategy, while effective within a certain range, cannot fully compensate for large-scale geometric variations.

The model's response to loading complexity represents another critical boundary condition. The high maximum errors observed in multiple loading cases serve as a warning sign, indicating that predictions for complex loading patterns should be approached with particular caution. This caution becomes especially important when dealing with stiffened structures, where the interaction between complex loading and stiffener configurations can lead to prediction errors significantly larger than those observed in simpler cases.

The presence of stiffeners introduces its own set of limitations. Our analysis reveals that certain stiffener configurations produce notably higher error rates and broader error distributions than others. This sensitivity to stiffener configuration becomes particularly acute under non-standard loading conditions, suggesting that additional validation measures may be necessary when applying the model to novel stiffener arrangements or unusual loading scenarios.

These limitations carry significant implications for the practical application of our framework. They suggest that successful implementation requires careful consideration of the intended use case and its relationship to the model's established reliability boundaries. In addition, they point to the potential value of developing confidence metrics that could help users identify when predictions approach these reliability boundaries. Such metrics could prove particularly valuable in cases where multiple complexity factors coincide, potentially amplifying prediction uncertainty.

5.4.5 Static Analysis Performance

While our framework demonstrated strong performance in buckling prediction, the results for static displacement field prediction were less competitive compared to state-of-the-art implementations like EA-GNN [11]. Our implementation achieved a relative validation error of 13.7% for displacement field prediction, notably higher than average 5% errors reported by Gladstone et al. This performance gap can be attributed to several factors.

First, our dataset generation and regularization efforts focused primarily on achieving uniform distribution of buckling eigenvalues, potentially at the expense of displacement field diversity. As identified by Yang et al. [54], regression tasks can suffer from data imbalance where certain value ranges are underrepresented, leading to biased models. While our regularization strategy proved effective for buckling prediction, a different approach focusing on displacement field distribution might be necessary for improved static analysis performance.

Second, the static displacement field represents a more complex prediction target than buckling eigenvalues, requiring accurate point-wise predictions across the entire domain rather than a single global value. The success of Gladstone et al.'s implementation suggests that their edge augmentation strategy may be particularly effective for capturing local deformation patterns, an aspect where our implementation could be improved.

These observations highlight the importance of tailored data preparation and architectural choices based on the specific prediction task. While our framework excels at capturing global structural behavior for buckling analysis, further refinements would be needed to match state-of-the-art performance in displacement field prediction.

5.5 Real Life Examples and Buckling Model Results

To evaluate the practical applicability of our Graph Neural Network framework, we conducted tests on two real-life stiffened panel configurations from aerospace applications. While our training dataset provided valuable insights into model behavior across various geometric configurations, testing on actual aerospace components offers a more clear evaluation of the model's generalization capabilities and practical utility.

5.5.1 Aircraft Bulkhead Analysis

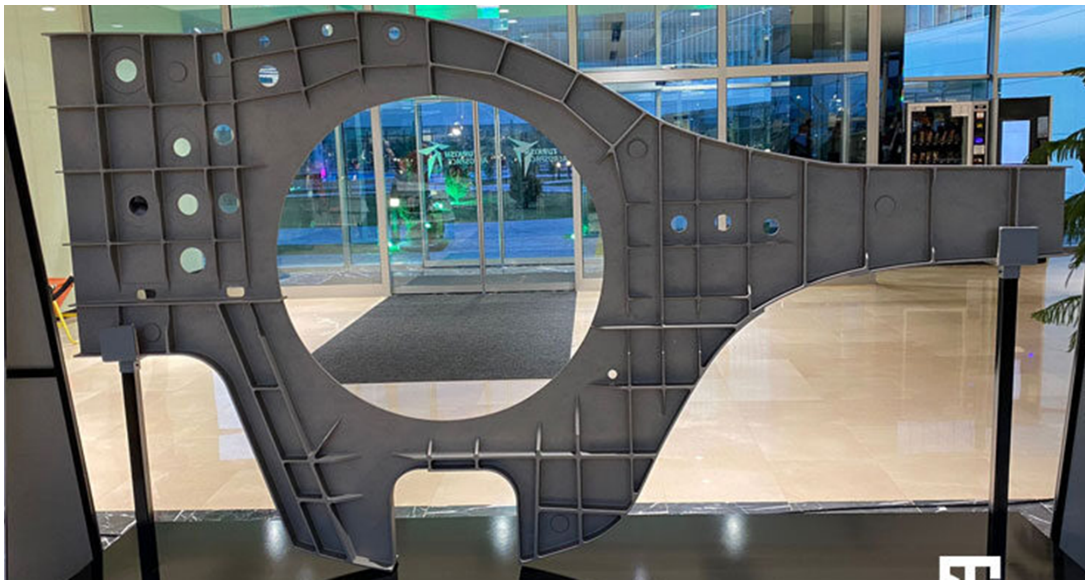
The first test case involved a fighter aircraft bulkhead featuring a large central cutout, representing a significantly more complex geometry than our training examples. The bulkhead size was scaled to the largest dimension of 1 meter, placing it at the upper bound of our training dataset's geometric range. Then, the bulkhead was modeled with fixed boundary conditions at its central region, with a 3kN force applied in the y-direction at the wing attachment region. Mesh size was kept 22mm, the same with training dataset. Figure 5.8 illustrates both the physical structure and its finite element representation.

Finite element analysis predicted a first buckling eigenvalue of 2.02, while our GNN-based model predicted 3.21, resulting in a relative error of 58.7%. This substantial deviation from FEA results highlights the challenges in generalizing to more complex, realistic geometries that deviate significantly from the training distribution.

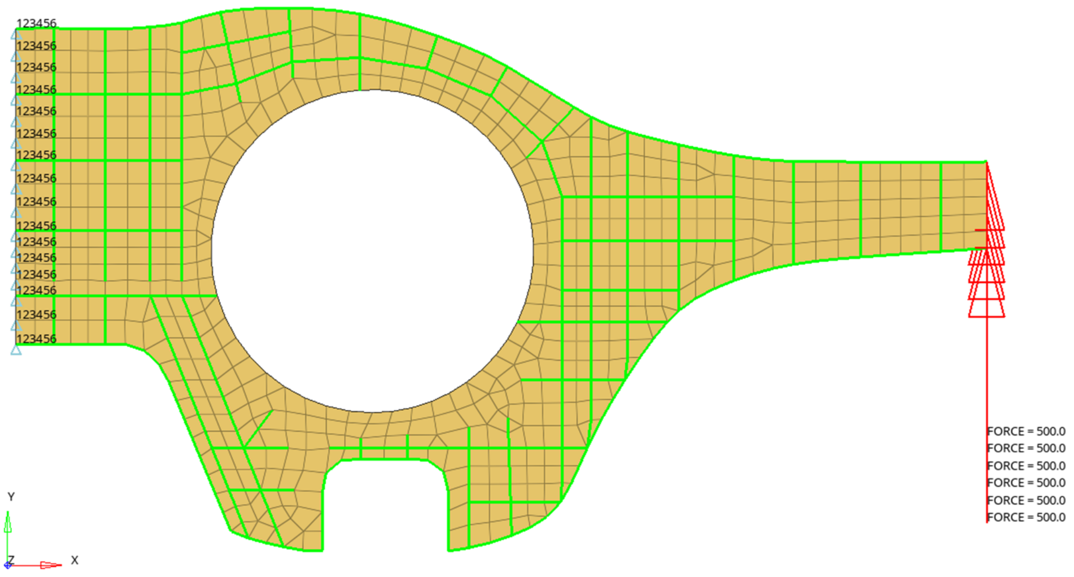
5.5.2 Aircraft Beam Analysis

The second test case examined an aircraft structural beam with a cutout, representing another common aerospace component. The beam was analyzed with fixed boundary conditions on one end and a 1.7kN force applied in the x-direction at the opposite end. The beam size was also scaled to have the largest dimension of 1 meter. The configuration is shown in Figure 5.9.

For this case, FEA predicted a buckling eigenvalue of 1.13, while our model predicted 1.43, representing a relative error of 26.9%. Better performance compared to the bulkhead case suggests that the beam's geometry and loading condition were more closely aligned with the scenarios represented in our training dataset.



(a) Physical structure [62]

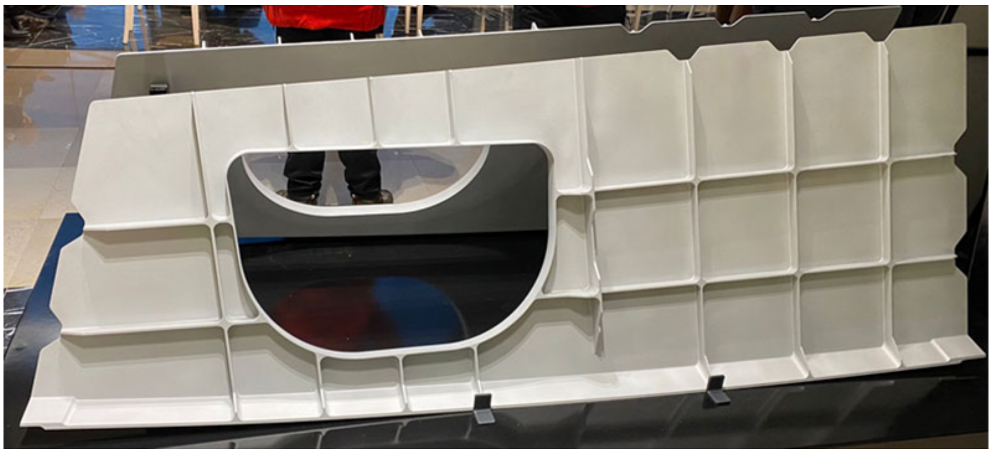


(b) Finite element model

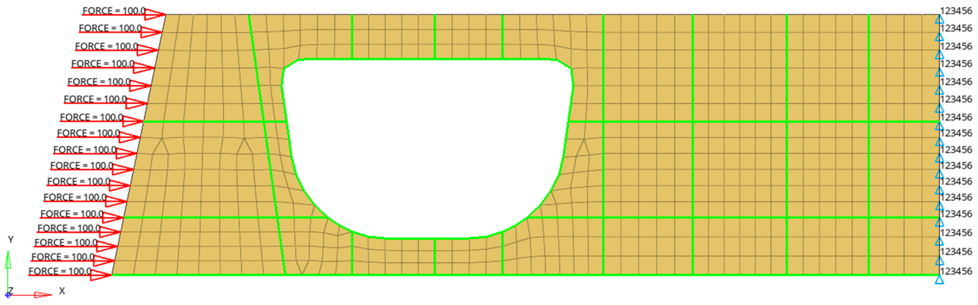
Figure 5.8: Fighter aircraft bulkhead analysis: (a) Physical structure [62] (b) Finite element model with boundary conditions and loading

5.5.3 Analysis of Results

These real-world test cases reveal several important insights about our model's current capabilities and limitations.



(a) Physical structure [62]



(b) Finite element model

Figure 5.9: Aircraft structural beam analysis: (a) Physical structure [62] (b) Finite element model showing boundary conditions and loading

In terms of geometric complexity, both test cases featured more intricate geometries than our training examples, particularly in terms of stiffener arrangements and cutout configurations. The higher prediction errors suggest that our training dataset may not adequately capture the geometric complexity encountered in actual aerospace components.

As for scale effects, the physical dimensions of these components often fall outside our training dataset's range (700mm to 1000mm) and aspect ratio range (0.5 to 1), potentially contributing to the reduced accuracy. This aligns with our earlier observations regarding the model's sensitivity to geometric scaling.

When it comes to loading conditions, the real-world loading scenarios, particularly the interaction between loads and complex geometric features like different cutout shapes, present challenges not fully represented in our training data. Also it should be noted that the boundary conditions in practical applications often involve more complex constraints than those used in training.

Finally, in terms of stiffener configurations, the actual arrangement and sizing of stiffeners in these components reflect practical design considerations that may differ significantly from our generated training examples. This suggests a need for more realistic stiffener patterns in our training dataset.

5.6 Comparative Analysis Between FEM and Surrogate Model

To evaluate the computational efficiency of our Graph Neural Network approach compared to traditional Finite Element Analysis, we conducted timing studies on both methods. The tests were performed on a workstation equipped with Intel Xeon 6148 2.40GHz CPU and NVIDIA Tesla V100 GPU. Each timing measurement represents the average of 100 consecutive runs to ensure statistical reliability.

For consistent comparison, we used a representative test case with 726 nodes and 4800 bidirectional edges, representing a typical structural configuration from our dataset. The GNN model employed was our optimized GraphSAGE architecture with 6 layers and a hidden dimension size of 512, configured for eigenvalue prediction. For the FEM analysis, we utilized MSC Nastran with both single-core and parallel processing modes, limiting CPU cores to 8 to maintain consistent comparison conditions.

Table 5.2: Performance Comparison Between GNN and FEM Analysis

Method	**Batch Size**	**Inference Time (s)**
GNN (CPU)	256	0.016
GNN (V100 GPU)	32	0.006
FEM (Single-core)	1	1.728
FEM (8-core Parallel)	1	0.522

The results demonstrate a significant performance advantage for the GNN-based approach. The CPU-accelerated GNN achieves an inference time of 16.1 milliseconds for a batch of 256 structures, translating to approximately 62.13 structures per second. In GPU-only operation, the GNN maintains impressive performance with 6.2 milliseconds per inference for a batch of 32 structures, processing about 160.45 structures per second.

In contrast, traditional FEM analysis using Nastran requires substantially more computational time. Single-core execution takes approximately 1.73 seconds per structure, while parallel processing with 8 cores reduces this to 0.52 seconds per structure. This translates to throughput rates of 0.58 and 1.92 structures per second respectively.

This performance comparison is not entirely fair because we could code FEM analysis to be run on CUDA and use the hardware advantage like GNN model. We could say it is more fairer to compare CPU-only performance of two model with 1 batch size. We calculated single core performance of Nastran as 1.73 seconds per structure. Even in this configuration, GNN model achieved 0.182 seconds per structure and showed its efficiency over traditional FEM analysis.

The performance difference becomes particularly significant when considering batch processing capabilities. While FEM analysis inherently processes one structure at a time, our GNN approach can efficiently handle multiple structures simultaneously, leading to dramatically higher throughput. This batch processing capability, combined with the inherent parallelization advantages of neural networks, results in approximately two orders of magnitude improvement in processing speed compared to traditional FEM methods.

These timing results highlight a key advantage of our GNN-based surrogate model approach: its ability to provide rapid structural analysis results while maintaining acceptable accuracy levels (as demonstrated in previous sections). This speed advantage becomes particularly valuable in scenarios requiring multiple analyses, such as design optimization or parametric studies, where the computational efficiency could significantly accelerate the design process.

This computational advantage becomes particularly significant when considering real-world aerospace applications. Consider the fuselage section shown in Figure 5.10, which consists of approximately 400 individual pocket panels of similar size to our test mesh (726 nodes, 4800 bidirectional edges). When designing such structures, engineers need to analyze multiple load cases - typically around 1000 different combinations of loads - to ensure structural integrity across all possible operating conditions.

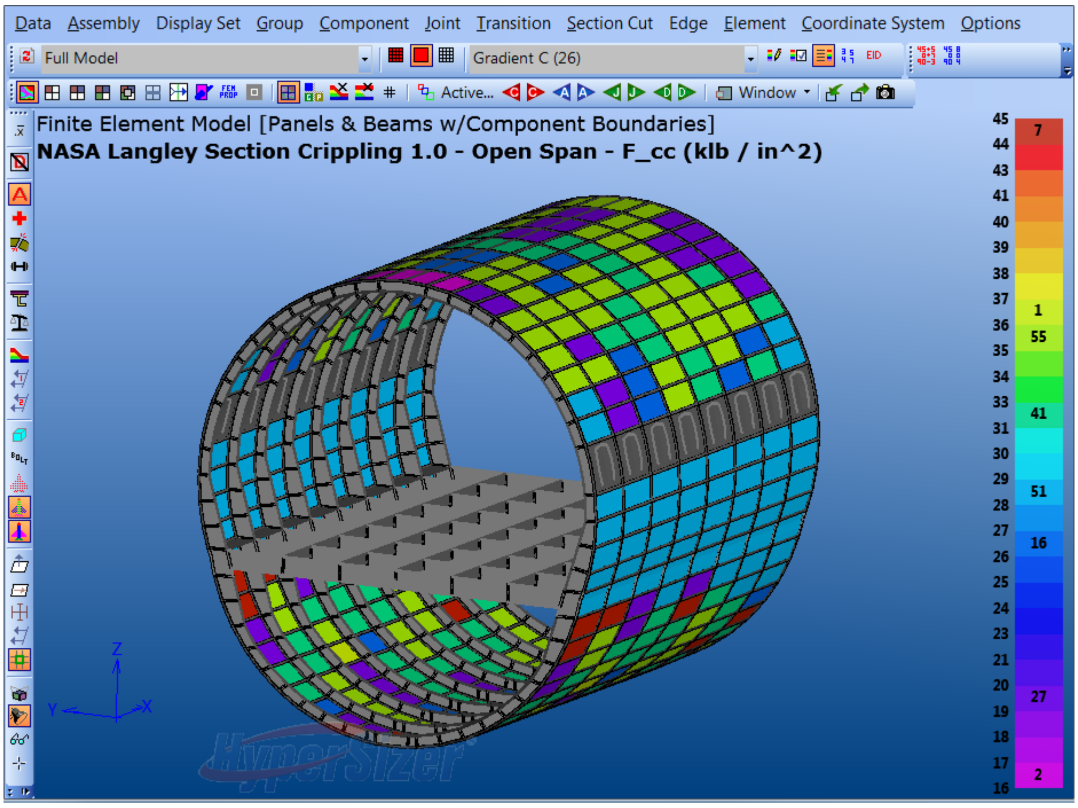


Figure 5.10: An aircraft fuselage section with multiple panels [63]

Let us quantify the computational requirements for analyzing this fuselage section. With traditional FEM analysis using 8-core parallel processing (0.522 seconds per analysis), analyzing all 400 panels under 1000 load cases would require:

$$T_{FEM} = 400 \text{ panels} \times 1000 \text{ load cases} \times 0.522 \text{ seconds} \approx 58.0 \text{ hours} \tag{5.2}$$

In contrast, our GNN model, leveraging GPU acceleration and batch processing capabilities (0.006 seconds for 32 structures), can complete the same analysis in:

$$T_{GNN} = 400 \text{ panels} \times 1000 \text{ load cases} \times 0.006 \text{ seconds} \approx 40.0 \text{ minutes} \tag{5.3}$$

This dramatic reduction in computation time - from over two days to less than half an hour - has profound implications for the aircraft design process. The design of aerospace structures is inherently iterative, requiring multiple rounds of analysis and optimization.

Each iteration traditionally requires a complete reanalysis of the structure under all load cases, making the design process time-consuming and computationally expensive. Our GNN-based surrogate model transforms this workflow by providing near-instantaneous feedback on structural performance.

The ability to reduce analysis time from days to minutes while maintaining acceptable accuracy levels (as demonstrated in previous sections) represents a paradigm shift in aerospace structural design. This efficiency gain not only accelerates the design process but also enables more thorough exploration of design alternatives, potentially leading to more optimized and innovative structural solutions. These timing improvements become even more significant when considering the entire aircraft design process, where multiple sections require similar analyses, and design changes in one area often necessitate re-analysis of adjacent sections. The computational efficiency of our GNN approach could potentially reduce the overall aircraft structural design cycle from weeks to days, providing a substantial competitive advantage in aerospace development programs.

5.7 Discussion and Analysis

The development and evaluation of our Graph Neural Network framework for structural analysis has provided valuable insights into both the capabilities and limitations of deep learning approaches in this domain. Our systematic investigation, progressing from simple non-stiffened structures to more complex stiffened configurations, reveals several key findings that have important implications for the future of computational structural analysis.

The most striking result from our study is the remarkable prediction accuracy achieved for buckling eigenvalues in non-stiffened structures, with validation MAPE of 5.5% and test MAPE of 6.73%. This level of accuracy, achieved while maintaining computational speeds orders of magnitude faster than traditional FEM analysis, demonstrates the viability of neural network-based surrogate models for practical structural analysis tasks. The small gap between validation and test performance indicates robust generalization capabilities, suggesting that our approach successfully captures fundamental relationships between geometric features, loading conditions, and buckling behavior.

However, the transition to stiffened structures revealed both the strengths and limitations of our approach. The increase in validation MAPE to 12.5% and test MAPE to 17.64% for stiffened structures reflects the inherent complexity added by stiffener elements. This performance degradation, while significant, remains within acceptable bounds for many practical applications, particularly in preliminary design phases where rapid analysis capabilities often outweigh the need for extreme precision. The larger gap between validation and test performance in stiffened structures (approximately 5 percentage points versus 1.2 for non-stiffened) suggests that the model faces greater challenges in generalizing learned patterns when stiffener-plate interactions are present.

Our investigation of different architectural components yielded several significant insights. The superior performance of GraphSAGE over CustomGNN architecture, particularly in capturing global structural behavior, highlights the importance of effective neighborhood aggregation in structural analysis tasks. The success of the super node approach, especially in buckling eigenvalue prediction, demonstrates the value of maintaining global structural information throughout the message passing process. This finding aligns with the physical nature of buckling phenomena, where local geometric features must be synthesized to predict global structural behavior.

The scale invariance tests revealed both the capabilities and limitations of our coordinate normalization strategy. While the model maintains reasonable performance across a range of scales, the systematic increase in prediction errors with scaling magnitude suggests that absolute size remains a relevant factor despite our normalization efforts. This scale sensitivity appears to be amplified by the presence of stiffeners, possibly due to the changing relative significance of stiffener-plate interactions at different scales. These findings suggest that future improvements in scale invariance might require more sophisticated normalization strategies or explicit incorporation of scale information in the model architecture.

Perhaps the most challenging aspect revealed by our study is the model's sensitivity to complex loading patterns, particularly in stiffened structures. The high maximum errors observed in multiple loading test cases (reaching 462.59% for stiffened structures) indicate potential stability issues in extreme scenarios. This sensitivity likely stems from the complex load paths that can develop through stiffener networks, creating structural responses that may deviate significantly from the patterns observed in the training data.

The contrast between our results in buckling prediction and the challenges encountered in displacement field prediction (where our implementation did not match the performance reported by Gladstone et al. [11]) provides interesting insights into the nature of different structural prediction tasks. The success in buckling prediction, despite using a simpler architecture, suggests that global structural properties might be more amenable to graph-based learning approaches than local field quantities. This observation aligns with the theoretical strengths of graph neural networks in capturing relationships across multiple scales.

Looking toward practical applications, our results suggest that this approach is particularly well-suited for preliminary design stages and optimization studies where rapid analysis capabilities are crucial. The dramatic reduction in computational time - from hours to minutes for typical aerospace structural components - could transform the design process, enabling more comprehensive exploration of design spaces and more thorough optimization studies. However, the identified limitations, particularly regarding scale invariance and complex loading conditions, indicate that these tools should be used with appropriate understanding of their boundaries and limitations.

The performance characteristics observed across different test scenarios also suggest potential directions for future improvement. The systematic patterns in prediction errors, particularly their relationship with geometric scale and loading complexity, indicate that more sophisticated feature engineering or architectural innovations might further enhance model robustness. Additionally, the success of the super node approach in capturing global structural behavior suggests that similar strategies for maintaining multi-scale structural information might benefit other prediction tasks.

In conclusion, while our framework demonstrates significant promise for accelerating structural analysis workflows, it also highlights the continuing challenges in creating truly general-purpose surrogate models for structural analysis. The balance between computational efficiency and prediction accuracy achieved in this work suggests that neural network-based approaches might best serve as complementary tools to traditional analysis methods, particularly in early design stages where rapid iteration is crucial. The insights gained from this study provide valuable direction for future research in this rapidly evolving field, pointing toward architectural innovations that might better capture the multi-scale nature of structural behavior.

CHAPTER 6

CONCLUSION

This research has advanced the application of Graph Neural Networks to structural analysis, particularly focusing on buckling behavior prediction. Through the development of a comprehensive framework encompassing data generation, graph representation and specialized neural network architectures, we have demonstrated both the potential and current limitations of deep learning approaches in structural engineering applications.

Our implementation achieved remarkable accuracy in buckling eigenvalue prediction for non-stiffened structures, with validation MAPE of 5.5% and test MAPE of 6.73%. This performance, combined with computational speeds approximately two orders of magnitude faster than traditional FEM analysis, represents a significant advancement in rapid structural analysis capabilities. For stiffened structures, while the accuracy decreased (validation MAPE 12.5%, test MAPE 17.64%), the results remain within acceptable bounds for many practical applications, particularly in preliminary design phases.

The research contributions span several key areas. First, our data generation pipeline, incorporating Bezier curve-based shape generation, mesh generation and systematic load case generation, provides a robust foundation for creating diverse, physically meaningful training datasets. The development of enhanced dataset balancing methodologies addresses the inherent biases in structural response distributions, ensuring more effective model training. Second, our investigation of different graph representation strategies, particularly the comparison between random virtual edges and super node architectures, provides valuable insights into effective information flow mechanisms for structural analysis tasks. The superior performance of the super node approach in buckling prediction highlights the importance of maintaining global structural information throughout the analysis process.

A particularly significant contribution lies in our comprehensive evaluation of model generalization capabilities. The systematic testing across scaled geometries (0.5× to 2.0×) and complex loading conditions provides clear boundaries for practical application. While the model maintains reasonable accuracy across various scales, the observed performance degradation with increasing scale deviation suggests limitations in our current approach to scale invariance. Similarly, the increased prediction errors under complex loading conditions, particularly for stiffened structures, indicate areas requiring further research attention.

From a practical perspective, our framework's most immediate impact lies in its potential to accelerate preliminary design processes. The ability to reduce analysis time from hours to minutes for typical aerospace structural components could transform design optimization workflows, enabling more comprehensive exploration of design spaces. However, the identified limitations in scale invariance and complex loading scenarios suggest that these tools should be viewed as complementary to traditional analysis methods rather than replacements.

In conclusion, while our research demonstrates significant progress in developing practical neural network-based surrogate models for structural analysis, it also highlights the continuing challenges in this field. The insights gained from this work, particularly regarding the importance of global information flow and the challenges of scale invariance, provide valuable direction for future research. As computational capabilities continue to advance and architectural innovations emerge, the integration of deep learning approaches with traditional structural analysis methods promises to enhance engineering design workflows substantially

The success of this research in dramatically reducing computational time while maintaining acceptable accuracy levels suggests that we are approaching a transformative moment in structural engineering practice. However, the thoughtful application of these tools, with clear understanding of their capabilities and limitations, will be crucial for their effective integration into engineering workflows. The path forward lies not in replacing traditional methods entirely, but in developing complementary approaches that leverage the strengths of both neural network-based and classical analysis techniques.

6.1 Future Works

This research opens several promising avenues for advancing machine learning in structural analysis and design optimization, focusing on both methodological enhancements and practical applications.

The development of more sophisticated scale invariance mechanisms could extend the framework's applicability across broader geometric ranges. Also, exploring advanced neural architectures, such as attention mechanisms or transformers, could better capture complex structural relationships. Multi-head attention and specialized super nodes focusing on distinct structural behaviors may enhance prediction accuracy for both global and local responses.

Integrating the surrogate model into genetic algorithm-based optimization offers immediate impact, significantly reducing computation times. Traditional optimization workflows could be accelerated from weeks to hours, enabling more comprehensive exploration of design spaces.

Beyond genetic algorithms, reinforcement learning frameworks could leverage our surrogate model for rapid evaluation in simultaneous optimization of stiffener layouts and panel geometries, unlocking vast design spaces and uncovering novel solutions. This method especially valuable because reinforcement learning can learn effective policies to increase buckling load capacity of structures using stiffeners. Trained reinforcement learning agents can optimize stiffener layouts of the structures in a short time.

Further improvements, such as scale-invariant normalization, physics-informed constraints, and uncertainty quantification, would enhance model stability and reliability, particularly for optimization applications.

These directions promise transformative changes in structural engineering, enabling faster, more informed decision-making and comprehensive design exploration. While requiring rigorous validation to ensure practical applicability, their potential to revolutionize design workflows makes them highly worthwhile for future research.

REFERENCES

[1] Stehan Timoshenko. Theory of plates and shells. *McGRAWC HILL*, 1959.

[2] J.C. Ekvall, J.E. Rhodes, and G.G. Wald. Methodology for evaluating weight savings from basic material properties. In *Design of fatigue and fracture resistant structures*. ASTM international, 1982.

[3] Klaus-Jürgen Bathe. *Finite element procedures*. Prentice Hall, 2006.

[4] Olek C. Zienkiewicz and Robert L. Taylor. *The finite element method for solid and structural mechanics*. Elsevier, 2005.

[5] Alexander I. Forrester and Andy J. Keane. Recent advances in surrogate-based optimization. *Progress in aerospace sciences*, 45(1-3):50–79, 2009.

[6] O.F. Hughes and J.K. Paik. *Ship Structural Analysis and Design*. The Society of Naval Architects and Marine Engineers, 2010.

[7] M. Papadrakakis, Nikos D. Lagaros, and Y. Tsompanakis. Structural optimization using evolution strategies and neural networks. *Computer methods in applied mechanics and engineering*, 156(1-4):309–333, 1998.

[8] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[9] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

[10] Yuan-Tung Chou, Wei-Tze Chang, Jimmy G. Jean, Kai-Hung Chang, Yin-Nan Huang, and Chuin-Shan Chen. Structgnn: An efficient graph neural network framework for static structural analysis. *Computers & Structures*, 299:107385, 2024.

[11] Rini Jasmine Gladstone, Helia Rahmani, Vishvas Suryakumar, Hadi Meidani, Marta D'Elia, and Ahmad Zareei. Mesh-based gnn surrogates for time-independent pdes. *Scientific reports*, 14(1):3394, 2024.

[12] Yuecheng Cai and Jasmin Jelovica. Efficient graph representation in graph neural networks for stress predictions in stiffened panels. *Thin-Walled Structures*, 203:112157, 2024.

[13] United States. Federal Aviation Administration, Battelle Memorial Institute. Columbus Laboratories, William J. Hughes Technical Center (U.S.), United States. Department of Defense, United States. National Aeronautics, and Space Administration. *MMPDS-13: Metallic Materials Properties Development and Standardization (MMPDS)*. MMPDS-13: Metallic Materials Properties Development and Standardization. Federal Aviation Administration, 2018.

[14] Robert D. Cook. *Concepts and applications of finite element analysis*. John wiley & sons, 2007.

[15] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[16] Jure Leskovec. Cs224w: Machine learning with graphs. `https://web.stanford.edu/class/cs224w/`, 2024. Stanford University, Fall 2024.

[17] William L. Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[18] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[19] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[21] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[22] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[23] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[24] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.

[25] Xuqiang Shao, Zhijian Liu, Siqi Zhang, Zijia Zhao, and Chenxing Hu. Pignn-cfd: A physics-informed graph neural network for rapid predicting urban wind field defined on unstructured mesh. *Building and Environment*, 232:110056, 2023.

[26] N.W. Murray. *Theory and Problems of Structural Steel Design*. McGraw-Hill, 1973.

[27] Elmer Franklin Bruhn. Analysis and design of flight vehicle structures. *(No Title)*, 1973.

[28] Hexagon AB. *MSC Nastran 2024.2 Linear Static Analysis User Guide*, 2024. Accessed: 2024-12-22.

[29] M. Muameleci. Linear and nonlinear buckling analyses of plates using finite element method. Ms thesis, Linköping University, Linköping, 2014.

[30] E. Avi, I. Lillemäe, J. Romanoff, and A. Niemelä. Equivalent shell element for ship structural design. *Ships and Offshore Structures*, 10(3):239–255, 2015.

[31] Patrick E. Fenner and Andrew Watson. Finite element buckling analysis of stiffened plates with filleted junctions. *Thin-Walled Structures*, 59:171–180, 2012.

[32] DG Stamatelos, GN Labeas, and KI Tserpes. Analytical calculation of local buckling and post-buckling behavior of isotropic and orthotropic stiffened panels. *Thin-Walled Structures*, 49(3):422–430, 2011.

[33] Z. Sun, Z. Lei, R. Bai, H. Jiang, J. Zou, Y. Ma, and C. Yan. Prediction of compression buckling load and buckling mode of hat-stiffened panels using artificial neural network. *Engineering Structures*, 242:112275, 2021.

[34] John C Houbolt and Elbridge Z Stowell. Critical stress of plate columns. Technical report, National Advisory Committee for Aeronautics, 1950.

[35] George E.P. Box, J. Stuart Hunter, and William G. Hunter. *Statistics for experimenters: design, innovation, and discovery*. Wiley-Interscience, 2005.

[36] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

[37] Rolland L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research*, 76(8):1905–1915, 1971.

[38] Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.

[39] K. Tian, Z. Li, X. Ma, H. Zhao, J. Zhang, and B. Wang. Transfer learning based variable-fidelity surrogate model for shell buckling prediction. *Structural and Multidisciplinary Optimization*, 61:1515–1528, 2021.

[40] V. Limbachiya and R. Shamass. Application of artificial neural networks for web-post shear resistance of cellular steel beams. *Thin-Walled Structures*, 161:107414, 2021.

[41] Navid Nourian, Mamdouh El-Badry, and Maziar Jamshidi. Design optimization of truss structures using a graph neural network-based surrogate model. *Algorithms*, 16(8):380, 2023.

[42] Rui Gao, Ishaan K. Deo, and Rajeev K. Jaiman. Node-element hypergraph message passing for fluid dynamics simulations. *arXiv preprint arXiv:2212.14545*, 2023.

[43] Shivam Barwey, Varun Shankar, and Romit Maulik. Multiscale graph neural network autoencoders for interpretable scientific machine learning. *arXiv preprint arXiv:2302.06186*, 2023.

[44] Ling-Han Song, Chen Wang, Jian-Sheng Fan, and Hong-Ming Lu. Elastic structural analysis based on graph neural network without labeled data. *Computer-Aided Civil and Infrastructure Engineering*, 38(10):1307–1323, 2023.

[45] Luca Pegolotti, Martin R. Pfaller, Natalia L. Rubio, Ke Ding, Rita Brugarolas Brufau, Eric Darve, and Alison L. Marsden. Learning reduced-order models for cardiovascular simulations with graph neural networks. *Computers in Biology and Medicine*, 168:107676, 2024.

[46] Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. *arXiv preprint arXiv:2210.00612*, 2022.

[47] Saurabh Deshpande, Jakub Lengiewicz, and Stéphane Bordas. Magnet: A graph u-net architecture for mesh-based simulations. *arXiv preprint arXiv:2211.00713*, 2022.

[48] Nicholas Black and Ahmad R. Najafi. Learning finite element convergence with the multi-fidelity graph neural network. *Computer Methods in Applied Mechanics and Engineering*, 397:115120, 2022.

[49] M. Lino, C. Cantwell, A.A. Bharath, and S. Fotiadis. Simulating continuum mechanics with multi-scale graph neural networks. *arXiv preprint arXiv:2106.04900*, 2021.

[50] Elissa Ross and Daniel Hambleton. Using graph neural networks to approximate mechanical response on 3d lattice structures. *Proceedings of AAG2020-Advances in Architectural Geometry*, 24:466–485, 2021.

[51] Mehdi Taghizadeh, Mohammad Amin Nabian, and Negin Alemazkoor. Multifidelity graph neural networks for efficient and accurate mesh-based partial differ-

ential equations surrogate modeling. *Computer-Aided Civil and Infrastructure Engineering*, 2024.

[52] Nan Liu, Yixuan Yu, Hanlin You, and Nikhil Tatikola. Ino: Invariant neural operators for learning complex physical systems with momentum conservation. *arXiv preprint arXiv:2212.14365*, 2022.

[53] Zhanghao Wu, Shuai Pan, Feng Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

[54] Yuzhe Yang, Kaiwen Zha, Yingcong Chen, Hao Wang, and Dina Katabi. Delving into deep imbalanced regression. In *International conference on machine learning*, pages 11842–11851. PMLR, 2021.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[56] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[57] Boris Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10), 2019.

[58] Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[59] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga,

et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[61] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[62] Savunma Sanayist. Milli muharip uçak'ın (mmu) ilk üretilen parçaları sergilendi. https://www.savunmasanayist.com/milli-muharip-ucakin-mmu-ilk-uretilen-parcalari-sergilendi/, 2022. Accessed: 2025-01-09.

[63] Collier Research Corporation. Hypersizer main page image. https://hypersizer.com. Accessed: 2025-01-04.

APPENDIX A

DATA AVAILABILITY

The complete implementation of the Graph Neural Network framework presented in this thesis, including all source code, datasets, and supplementary materials, is publicly available in an open-source repository. This repository contains:

- Source code for shape generation, data preprocessing, and model implementation
- Sample datasets for demonstration and validation
- Documentation and usage instructions
- Utility scripts for data processing and visualization

The repository can be accessed at:

`https://github.com/omerkurt-okt/buck-gnn`

All code is released under the MIT License, allowing for both academic and commercial use. We encourage the research community to build upon this work and welcome contributions for further improvements.

For any technical issues or questions regarding the implementation, users can submit inquiries through the repository's issue tracking system. This approach to open-source distribution aligns with principles of reproducible research and facilitates future developments in the field of Graph Neural Networks for structural analysis.