Fachhochschule Südwestfalen
University of Applied Sciences

MASTER THESIS

Architecture Development to Replace Finite Element Methods with Deep Neural Networks

Examiners
Prof. Dr. – Ing. Andreas Schwung
&
Prof. Dr. – Ing. Jens Oberrath

Master Thesis in collaboration with
South Westphalia University of Applied Sciences
Soest, Germany

Advisor:
MSc. Johannes Pöppelbaum

Submitted by
Kaushal Narendra Tare
Matriculation No. 10062903
Course: M.Sc. Systems Engineering and Engineering Management
Submitted on: June 4, 2021

Declaration of Authorship

I, Kaushal Narendra Tare, hereby declare that:

- I am the sole author and composer of my thesis titled, "Architecture Development to Replace Finite Element Methods with Deep Neural Networks".
- The whole work was done while being the candidate for the master's degree at this University.
- No other sources or learning aids, other than those listed have been used.
- I have acknowledged the work of others by providing detailed references of the said work.
- My Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Signed: _____

Date: _____

Abstract

Ever since the industrial revolution, the focus has been to improve the manufacturing techniques by implementing new technologies in order to meet the demands of customers. Design and analysis being the building blocks of manufacturing processes, have transformed a lot, making the traditional time consuming manual work redundant over the years. This was possible because of the evolution of computers with high computing power and memory capacity. The design and analysis tasks which today are almost entirely solved using computer software packages are now taking a step further with customization and automation thus integrating with the concepts of 'Industry 4.0'.

Simulations being one of the major driving technologies of Industry 4.0, a large amount of simulation data is getting generated over the years. Machine Learning techniques which thrive on data can thus be used to exploit the information from past results and give solution to a new problem case. With the upward hike in computing power over the past couple of decades it is now possible to work with data on a large scale with various deep learning algorithms. The combination of large data generation and the ability to process it, have created the right environment to motivate the research of ML and specially Deep Neural Networks (DNN) in the field of design and simulation. One of the main advantages of ML based approach would be- avoiding the hours or days of simulation time for a new problem case and facilitating quick deployment in manufacturing suitable for todays dynamic fast paced production environment.

This thesis creates a framework that involves implementation of Graph Neural Networks (GNNs) to give solution of a simulation problem case scenario. The case study selected for this purpose is that of a cantilever beam subjected to loading conditions (forces as inputs) and giving output in form of deformations and induced stresses. Different DNN models with variation in graph network architectures such as Graph Convolutional Network (GCN) and Gated Graph Neural Network (GG-NN) will be implemented to test the prediction accuracy and thus compare with the results from simulation software to determine the effectiveness.

Based on the analysis of results obtained from experimenting GCN and GG-NN models on three datasets namely- the coarse mesh dataset, the finer mesh dataset and the twist case dataset, it is concluded that the GCN model outperforms GG-NN model in terms of level of accuracy in predicting the deformation and stress values. The GG-NN model performs good on coarse mesh dataset and finer mesh dataset but gives below par performance on twist case dataset, therefore making the GCN model a suitable generic choice for obtaining the best output predictions comparable to the FEA simulation results.

Acknowledgements

I would like to firstly thank Prof.Dr.-Ing Andreas Schwung for giving me an opportunity to work on this master thesis topic from my field of interest. He helped me with his ideas and valuable suggestions during the initial tough phase of my thesis journey which helped me build the required knowledge base for the further research. I would like to thank Prof.Dr.-Ing Jens Oberrath for examining my thesis.

I would also like to thank MSc. Johannes Pöppelbaum for his constant guidance and supervision which motivated me to give my best. The patience and support he showed when I was stuck due to shortage of ideas or various other difficulties was commendable.

Furthermore, I would like to thank MSc.Mohammed Sharafath, MSc. Kaustubh Shrotri, MSc. Utkarsh Panara, MSc. Prajakta Lokhande, Mr. Avadhoot Adawale and Mr. Parag Patil who were always there to hear my ideas and give useful advice which helped me clear various obstacles in my thesis journey. Finally, I would like to thank my family and friends who provided me with all the moral support required to keep me positive and motivated.

Contents

Declaration of Authorship iii

Abstract v

Acknowledgements vii

1 Introduction 1
1.1 Motivation and Aim of Work . . . 2
1.2 Computer Aided Engineering . . . 3
1.2.1 Finite Element Analysis . . . 4
1.2.2 Finite Element Method . . . 4
1.2.3 Scope and Limitations of FEM . . . 5
1.3 Machine Learning Approach . . . 5
1.4 Scope of Work . . . 6

2 Literature Review 7

3 Fundamentals of Finite Element Method 9
3.1 Strong Formulation of Continuous System . . . 9
3.2 Weak Formulation and Finite Element Solution . . . 11

4 Deep Learning 21
4.1 Artificial Neural Networks . . . 21
4.1.1 Neuron . . . 22
4.1.2 Activation Function . . . 22
4.1.3 Working of an Artificial Neural Network . . . 24
4.1.4 Gradient Descent . . . 25
4.1.5 Stochastic Gradient Descent . . . 26
4.1.6 Backpropagation . . . 27
4.2 Graph Neural Networks . . . 28
4.2.1 What is a Graph? . . . 28
4.2.2 Fundamentals of GNN . . . 29
Gather Locality Information . . . 30
Aggregate and Compute the Information . . . 31
4.2.3 General GNN Pipeline . . . 33
Defining the graph structure . . . 33
Specifying the graph type . . . 33
Designing the loss function . . . 34
Using computational modules and building the model . . . 34

5 Case Study Scenario **37**
5.1 Cantilever Beam . . . 37
5.2 Finite Element Analysis of Cantilever Beam . . . 38
5.2.1 Design of Cantilever Beam . . . 38
5.2.2 Generating Finite Element Mesh . . . 39
5.2.3 Specifying the Boundary Conditions . . . 39
5.2.4 Obtaining the Results . . . 40
5.2.5 Dataset Generation . . . 41
5.3 DNN Adaptation to Cantilever Beam Scenario . . . 42
5.3.1 Generating the Graph . . . 42
5.3.2 Building the Neural Network Model . . . 43

6 Experiments and Results **45**
6.1 Beam with Coarse Mesh . . . 45
6.1.1 Analysis with GCN Architecture for Coarse Mesh . . . 46
6.1.2 Analysis with Gated Graph Architecture for Coarse Mesh . . . 60
6.1.3 Effect of Shared Network vs. Independent Network . . . 69
6.1.4 Varying the Number of Layers in Neural Network . . . 70
6.1.5 Effect of Different Activation Functions . . . 71
6.2 Beam with Finer Mesh . . . 72
6.2.1 Analysis with GCN Architecture for Finer Mesh . . . 72
6.2.2 Analysis with Gated Graph Architecture for Finer Mesh . . . 80
6.3 Beam with Twisting . . . 87
6.3.1 Analysis with GCN Architecture for Twist Case . . . 88
6.3.2 Analysis with Gated Graph Architecture for Twist Case . . . 98
6.4 Summary and Discussion . . . 105

7 Conclusion and Future Scope **109**

A Appendix A - Nodal Deformation and Stress Plots **111**
A.1 Gated Graph Architecture on Coarse Mesh . . . 111
A.2 GCN Architecture on Finer Mesh . . . 115
A.3 Gated Graph Architecture on Finer Mesh . . . 120
A.4 GCN Architecture on Twist Case . . . 124
A.5 Gated Graph Architecture on Twist Case . . . 129

List of Figures

1.1 Product Development Cycle [1] . . . 1
1.2 Discretization [2] . . . 4

3.1 A continuous system of axially loaded bar . . . 9
3.2 Different types of finite elements . . . 11
3.3 Finite elements and shape functions for axially loaded bar . . . 13

4.1 Biological neuron vs. a perceptron [3] . . . 22
4.2 Perceptron with activation function . . . 23
4.3 Different activation functions[4] . . . 23
4.4 Basic working principle of ANN[4] . . . 24
4.5 Gradient Descent method[5] . . . 25
4.6 Batch gradient descent giving local minimum[5] . . . 26
4.7 A Graph . . . 28
4.8 Social network and a molecule as a graph[6] . . . 29
4.9 Node Embedding . . . 30
4.10 Gathering locality information[7] . . . 30
4.11 Input graph and a node computational graph[7] . . . 31
4.12 GNN deep model[7] . . . 32
4.13 General GNN design pipeline[6] . . . 33

5.1 A cantilever beam . . . 37
5.2 Ansys design modeller interface . . . 38
5.3 Meshing . . . 39
5.4 Boundary conditions . . . 40
5.5 Output directional deformation and equivalent stress . . . 41
5.6 Dataset structure . . . 42
5.7 Generating graph using integer label for nodes . . . 42
5.8 Graph network . . . 43
5.9 DNN model flowchart . . . 44

6.1 Cantilever beam with coarse meshing . . . 45
6.2 Model with GCN architecture . . . 46
6.3 Training and validation curve for model with GCN architecture . . . 47
6.4 Selection of most likely result given by model on test data . . . 47
6.5 An example of DNN result visualization for deformation . . . 48
6.6 FEM vs. GCN result for deformation at time step 1 . . . 48
6.7 FEM vs. GCN result for deformation at time step 2 . . . 48
6.8 FEM vs. GCN result for deformation at time step 3 . . . 49
6.9 FEM vs. GCN result for deformation at time step 4 . . . 49

6.10 FEM vs. GCN result for deformation at time step 5 49
6.11 Nodal deformations for time step 1 50
6.12 Nodal deformations for time step 2 50
6.13 Nodal deformations for time step 3 50
6.14 Nodal deformations for time step 4 51
6.15 Nodal deformations for time step 5 51
6.16 Error and relative error for deformation at time step 1 52
6.17 Error and relative error for deformation at time step 2 52
6.18 Error and relative error for deformation at time step 3 52
6.19 Error and relative error for deformation at time step 4 53
6.20 Error and relative error for deformation at time step 5 53
6.21 FEM vs. GCN result for stress at time step 1 54
6.22 FEM vs. GCN result for stress at time step 2 54
6.23 FEM vs. GCN result for stress at time step 3 54
6.24 FEM vs. GCN result for stress at time step 4 55
6.25 FEM vs. GCN result for stress at time step 5 55
6.26 Nodal stresses for time step 1 56
6.27 Nodal stresses for time step 2 56
6.28 Nodal stresses for time step 3 57
6.29 Nodal stresses for time step 4 57
6.30 Nodal stresses for time step 5 58
6.31 Error and relative error for stress at time step 1 58
6.32 Error and relative error for stress at time step 2 59
6.33 Error and relative error for stress at time step 3 59
6.34 Error and relative error for stress at time step 4 59
6.35 Error and relative error for stress at time step 5 60
6.36 Model with gated graph architecture 61
6.37 Training and validation curve for model with GG-NN architecture . . 61
6.38 FEM vs. GG-NN result for deformation at time step 1 61
6.39 FEM vs. GG-NN result for deformation at time step 2 62
6.40 FEM vs. GG-NN result for deformation at time step 3 62
6.41 FEM vs. GG-NN result for deformation at time step 4 62
6.42 FEM vs. GG-NN result for deformation at time step 5 63
6.43 Error and relative error for deformation at time step 1 63
6.44 Error and relative error for deformation at time step 2 64
6.45 Error and relative error for deformation at time step 3 64
6.46 Error and relative error for deformation at time step 4 64
6.47 Error and relative error for deformation at time step 5 65
6.48 FEM vs. GG-NN result for stress at time step 1 66
6.49 FEM vs. GG-NN result for stress at time step 2 66
6.50 FEM vs. GG-NN result for stress at time step 3 66
6.51 FEM vs. GG-NN result for stress at time step 4 67
6.52 FEM vs. GG-NN result for stress at time step 5 67
6.53 Error and relative error for stress at time step 1 68
6.54 Error and relative error for stress at time step 2 68
6.55 Error and relative error for stress at time step 3 68
6.56 Error and relative error for stress at time step 4 69

6.57 Error and relative error for stress at time step 5 69
6.58 Independent network and shared network architectures 70
6.59 Cantilever beam with finer meshing 72
6.60 Training and validation curve for model with GCN architecture 72
6.61 FEM vs. GCN result for deformation at time step 1 73
6.62 FEM vs. GCN result for deformation at time step 2 73
6.63 FEM vs. GCN result for deformation at time step 3 73
6.64 FEM vs. GCN result for deformation at time step 4 74
6.65 FEM vs. GCN result for deformation at time step 5 74
6.66 Error and relative error for deformation at time step 1 75
6.67 Error and relative error for deformation at time step 2 75
6.68 Error and relative error for deformation at time step 3 75
6.69 Error and relative error for deformation at time step 4 76
6.70 Error and relative error for deformation at time step 5 76
6.71 FEM vs. GCN result for stress at time step 1 76
6.72 FEM vs. GCN result for stress at time step 2 77
6.73 FEM vs. GCN result for stress at time step 3 77
6.74 FEM vs. GCN result for stress at time step 4 77
6.75 FEM vs. GCN result for stress at time step 5 78
6.76 Error and relative error for stress at time step 1 78
6.77 Error and relative error for stress at time step 2 79
6.78 Error and relative error for stress at time step 3 79
6.79 Error and relative error for stress at time step 4 79
6.80 Error and relative error for stress at time step 5 80
6.81 Training and validation curve for model with GG-NN architecture . . 80
6.82 FEM vs. GG-NN result for deformation at time step 1 81
6.83 FEM vs. GG-NN result for deformation at time step 2 81
6.84 FEM vs. GG-NN result for deformation at time step 3 81
6.85 FEM vs. GG-NN result for deformation at time step 4 82
6.86 FEM vs. GG-NN result for deformation at time step 5 82
6.87 Error and relative error for deformation at time step 1 82
6.88 Error and relative error for deformation at time step 2 83
6.89 Error and relative error for deformation at time step 3 83
6.90 Error and relative error for deformation at time step 4 83
6.91 Error and relative error for deformation at time step 5 84
6.92 FEM vs. GG-NN result for stress at time step 1 84
6.93 FEM vs. GG-NN result for stress at time step 2 84
6.94 FEM vs. GG-NN result for stress at time step 3 85
6.95 FEM vs. GG-NN result for stress at time step 4 85
6.96 FEM vs. GG-NN result for stress at time step 5 85
6.97 Error and relative error for stress at time step 1 86
6.98 Error and relative error for stress at time step 2 86
6.99 Error and relative error for stress at time step 3 86
6.100 Error and relative error for stress at time step 4 87
6.101 Error and relative error for stress at time step 5 87
6.102 Training and validation curve for Z-deformation and Stress with GCN architecture 88

6.103Training and validation curve for X and Y deformation with GCN architecture 88
6.104FEM vs. GCN result for deformation at time step 1 89
6.105FEM vs. GCN result for deformation at time step 2 89
6.106FEM vs. GCN result for deformation at time step 3 89
6.107FEM vs. GCN result for deformation at time step 4 90
6.108FEM vs. GCN result for deformation at time step 5 90
6.109Error and relative error for Z deformation at time step 1 90
6.110Error and relative error for Z deformation at time step 2 91
6.111Error and relative error for Z deformation at time step 3 91
6.112Error and relative error for Z deformation at time step 4 91
6.113Error and relative error for Z deformation at time step 5 92
6.114Error for X and Y deformation at time step 1 92
6.115Error for X and Y deformation at time step 2 93
6.116Error for X and Y deformation at time step 3 93
6.117Error for X and Y deformation at time step 4 93
6.118Error for X and Y deformation at time step 5 94
6.119FEM vs. GCN result for stress at time step 1 94
6.120FEM vs. GCN result for stress at time step 2 94
6.121FEM vs. GCN result for stress at time step 3 95
6.122FEM vs. GCN result for stress at time step 4 95
6.123FEM vs. GCN result for stress at time step 5 95
6.124Error and relative error for stress at time step 1 96
6.125Error and relative error for stress at time step 2 96
6.126Error and relative error for stress at time step 3 96
6.127Error and relative error for stress at time step 4 97
6.128Error and relative error for stress at time step 5 97
6.129Training and validation curve for Z-deformation and Stress with GG-NN architecture 98
6.130Training and validation curve for X and Y deformation with GG-NN architecture 98
6.131FEM vs. GG-NN result for deformation at time step 1 99
6.132FEM vs. GG-NN result for deformation at time step 2 99
6.133FEM vs. GG-NN result for deformation at time step 3 99
6.134FEM vs. GG-NN result for deformation at time step 4 100
6.135FEM vs. GG-NN result for deformation at time step 5 100
6.136Error and relative error for Z deformation at time step 1 100
6.137Error and relative error for Z deformation at time step 2 101
6.138Error and relative error for Z deformation at time step 3 101
6.139Error and relative error for Z deformation at time step 4 101
6.140Error and relative error for Z deformation at time step 5 102
6.141FEM vs. GG-NN result for stress at time step 1 102
6.142FEM vs. GG-NN result for stress at time step 2 102
6.143FEM vs. GG-NN result for stress at time step 3 103
6.144FEM vs. GG-NN result for stress at time step 4 103
6.145FEM vs. GG-NN result for stress at time step 5 103
6.146Error and relative error for stress at time step 1 104

6.147Error and relative error for stress at time step 2 104
6.148Error and relative error for stress at time step 3 104
6.149Error and relative error for stress at time step 4 105
6.150Error and relative error for stress at time step 5 105

A.1 Nodal deformations for time step 1 111
A.2 Nodal deformations for time step 2 111
A.3 Nodal deformations for time step 3 112
A.4 Nodal deformations for time step 4 112
A.5 Nodal deformations for time step 5 112
A.6 Nodal stresses for time step 1 113
A.7 Nodal stresses for time step 2 113
A.8 Nodal stresses for time step 3 114
A.9 Nodal stresses for time step 4 114
A.10 Nodal stresses for time step 5 115
A.11 Nodal deformations for time step 1 115
A.12 Nodal deformations for time step 2 116
A.13 Nodal deformations for time step 3 116
A.14 Nodal deformations for time step 4 116
A.15 Nodal deformations for time step 5 117
A.16 Nodal stresses for time step 1 117
A.17 Nodal stresses for time step 2 118
A.18 Nodal stresses for time step 3 118
A.19 Nodal stresses for time step 4 119
A.20 Nodal stresses for time step 5 119
A.21 Nodal deformations for time step 1 120
A.22 Nodal deformations for time step 2 120
A.23 Nodal deformations for time step 3 120
A.24 Nodal deformations for time step 4 121
A.25 Nodal deformations for time step 5 121
A.26 Nodal stresses for time step 1 122
A.27 Nodal stresses for time step 2 122
A.28 Nodal stresses for time step 3 123
A.29 Nodal stresses for time step 4 123
A.30 Nodal stresses for time step 5 124
A.31 Nodal Z deformations for time step 1 124
A.32 Nodal Z deformations for time step 2 125
A.33 Nodal Z deformations for time step 3 125
A.34 Nodal Z deformations for time step 4 125
A.35 Nodal Z deformations for time step 5 126
A.36 Nodal stresses for time step 1 126
A.37 Nodal stresses for time step 2 127
A.38 Nodal stresses for time step 3 127
A.39 Nodal stresses for time step 4 128
A.40 Nodal stresses for time step 5 128
A.41 Nodal Z deformations for time step 1 129
A.42 Nodal Z deformations for time step 2 129
A.43 Nodal Z deformations for time step 3 130

A.44 Nodal Z deformations for time step 4 130
A.45 Nodal Z deformations for time step 5 130
A.46 Error for X and Y deformation at time step 1 131
A.47 Error for X and Y deformation at time step 2 131
A.48 Error for X and Y deformation at time step 3 132
A.49 Error for X and Y deformation at time step 4 132
A.50 Error for X and Y deformation at time step 5 132
A.51 Nodal stresses for time step 1 133
A.52 Nodal stresses for time step 2 134
A.53 Nodal stresses for time step 3 134
A.54 Nodal stresses for time step 4 135
A.55 Nodal stresses for time step 5 135

List of Tables

6.1 Effect of number of layers on the prediction quality for coarse mesh dataset 71
6.2 Effect of activation function in MLP layers on the model performance for coarse mesh dataset 71
6.3 Effect of activation functions in GNN layers on the model performance for coarse mesh dataset 71
6.4 Model performances on deformation prediction 106
6.5 Model performances on stress prediction 106
6.6 Overall performance of models 107

Chapter 1

Introduction

Before the advent of computers, product development and manufacturing process involved the traditional techniques like drafting, lengthy hand-calculations and physical testing of prototype. After the 1960's, these techniques gradually got replaced by computer-based modelling and simulation. With time, this technology significantly expanded its capabilities because of more powerful processors, increased data storage capacities, improved visualization techniques and more functionality across widely spread platforms. Today, modelling and simulation is conducted on wide variety of Computer Aided Design (CAD) and Computer Aided Engineering (CAE) software packages [8]. Presently, we are in the middle of a significant transformation that is changing the way we manufacture products, all thanks to the increasing automation in the field of manufacturing. The digital revolution has completely changed traditional manufacturing processes, and we now have smart factories, virtual prototypes and intelligent machines. This digital revolution in manufacturing industry is tagged under the concept of "Industry 4.0" [9][10]. "Industry 4.0 refers to the intelligent networking of machines and processes for industry with the help of information and communication technology"[11]. According to [12] Computer Aided Engineering is one of the major technologies governing the Industry 4.0 concept. And even though CAE is one of the major technology involved in driving the Industry 4.0 concept, its methods are still mostly based on traditional techniques. An iterative product development process is shown in figure 1.1.

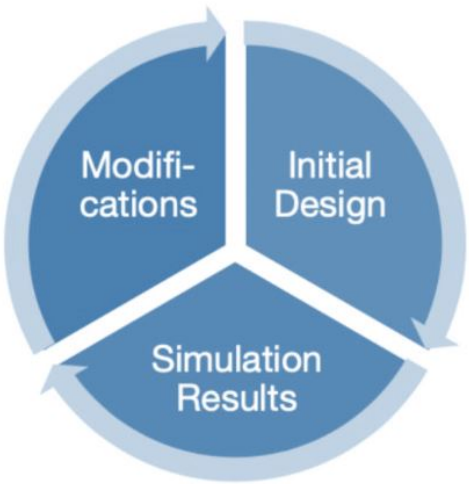

FIGURE 1.1: Product Development Cycle [1]

While this process is quick and efficient for simple scenarios or analysis of small components or assemblies, it tends to get computationally more expensive for larger, complex and detailed scenarios. The term 'computationally expensive' here means hours or even days of simulations, and since the simulation needs to re-run each time even for a small change, means that the design process gets slower and costly[13]. Over the years, the Finite Element (FE) Simulations have been improving by integrating more methods to solve complex and wide range of problems. In parallel, Artificial Intelligence (AI) and Machine Learning (ML) have also been advancing and making tremendous progress by inventing new methods that address the problems same as those in FE simulations [14].

1.1 Motivation and Aim of Work

Engineers often have to work on difficult tasks or projects, and there is always an uncertainty of failure associated with each task or project. Modern modelling and simulation techniques have greatly helped to reduce this risk factor. Today these techniques have become an integral part of manufacturing processes and product life cycle, hence there has always been the need to improve these techniques further and make them adaptable to future requirements or improve the quality of existing techniques to give better results.

The traditional physics based models or scenarios are based on governing equations which are partial different equations describing the physical phenomenon. However, with the access to Big Data generated with the help of sensors and other IoT devices, it has now become possible to train Machine Learning models that yield predictive simulation models[15]. Recent developments in Machine Learning involving implementation of new methods with the help of specially designed platforms running on GPU's are allowing the machine learning models to summarize the simulation results and cut short or provide an alternative to simulation process. The majority of new methods or modeling techniques that have brought most of the development collectively lie under 'Deep Learning'. Deep Learning is a subset of Machine Learning and a special method of information processing that uses multi-layered neural networks to analyze data and give output in required form. The rapid growth of Deep Learning techniques has now allowed us to tackle high complexity problems and simultaneously achieve high accuracy by being able to use large training datasets in an efficient manner. This has promoted the application of Deep Learning models in the field of simulation where we have complex problems of representing simulation results of Finite Element Analysis (FEA) used for product design and manufacturing on an industrial scale[16][14].

This thesis portrays a framework using a problem case scenario of a cantilever beam under loading condition that displays collection of data from FEA software, training the Deep Neural Network models on the collected data and obtaining comparable results to that from the FEA. The primary aim of this thesis is to find out whether the Deep Learning techniques are able to simulate an actual physical phenomenon, in this case the Cantilever Beam scenario. It would be interesting to find whether, the Deep Neural Network model can recognize the structural information of the cantilever beam just like modelling the geometry in FEA, whether it can process the input information of forces applied on cantilever beam, and whether it

learns efficiently to give the output(comparable to output obtained from FEA) in form of deformation and stress values at different locations of the cantilever beam.

1.2 Computer Aided Engineering

Computer Aided Engineering is the use of computer software to simulate the performance in order to improve product design or assist in the resolution of engineering problems in wide range of industries. The task domain of CAE involves simulation, optimization of products, and validation of products or manufacturing processes[17].

CAE normally consists of pre-processing, solving and post-processing steps. The pre-processing phase involves modelling the geometry, the physical properties and the environment in form of applied loads or constraints. The solving phase involves solving the model by using mathematical formulation following the underlying physics for the defined model. Finally, in the post-processing phase, the results are presented to the engineer for review[17]. From [18] some of the important engineering areas covered by CAE include,

- Stress Analysis and dynamics analysis of parts or assemblies with the help of Finite Element Analysis (FEA)
- Thermal and Fluid Analysis using Computational Fluid Dynamics(CFD)
- Multibody dynamics involving kinematics and dynamic analysis of mechanisms in Mechannical Event Simulations
- Control Systems Analysis
- Simulation of some important manufacturing prosses for eg. casting, molding and die press forming

According to [19] and [18] the important benefits of CAE are as follows,

- Reduced product development cost and time, with improved product quality and durability
- Design of product can be implemented, evaluated and improved and thus some important design decisions can be made based on the performance
- Simulation based design replacing the physical prototype testing saves lot of money and time
- CAE helps engineering teams manage risk and understand the performance implications of their designs
- Combined CAE and process management make it possible to effectively improve performance and leverage designs to a broader application
- When integrated with product design and manufacturing, CAE can enable earlier problem detection and thus help reduce maintenance cost associated with product life cycle

Because of wide functionality and various benefits, CAE has found its footing in various industries like automotive, defence and aerospace, electronics, medical devices and industrial equipment[20].

1.2.1 Finite Element Analysis

The study of behaviour of components in real time conditions in CAE is achieved through Finite Element Analysis. Finite Element Analysis is a computing technique used to simulate a physical phenomenon to obtain approximate solutions of boundary value problems (component or assembly under given boundary conditions). To achieve this, it uses numerical method called as Finite Element Method (FEM) [21][22].

1.2.2 Finite Element Method

Any physical phenomenon such as structural behaviour, fluid behaviour, thermal transport etc. can be understood and quantified with the help of mathematics. Most of the physical processes occurring in nature can be described with the help of Partial Differential Equations(PDEs) which are complicated equations that need to be solved in order to compute relevant required quantities such as deformation, stress, strain etc. This method considers the system to be a continuous domain of infinite elements and such systems are called as continuous systems. On the other hand, for computer to solve these PDEs, different numerical techniques have been developed and the most prominent one is the FEM. One of the important fact to note is that FEM gives approximate solution to a problem and is a numerical method to get the results of the PDE [22].

In FEM, the shape of structure being assessed is divided into many smaller elements creating a mesh. By doing this the continuous system is converted into a discrete system containing finite number of elements. This process is known as discretization and can be observed in figure 1.2.

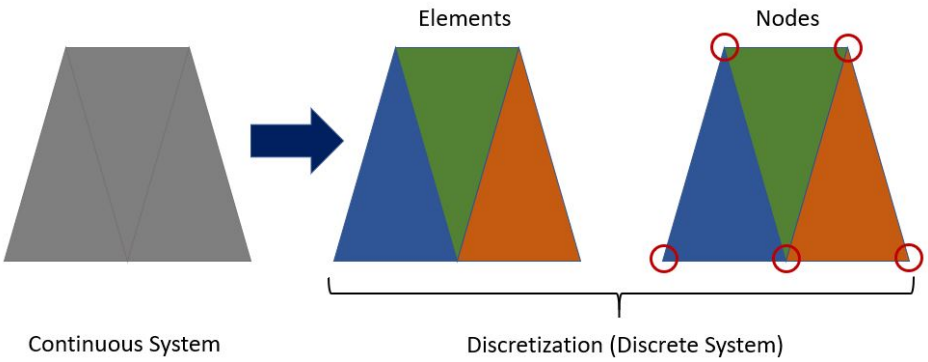

FIGURE 1.2: Discretization [2]

Calculations are done for each element and the individual results are combined to give the overall final result. These approximate calculations are usually polynomials and interpolations over the elements. Interpolating over the elements implies that we would only know the values at certain points but not all of them. The points

where the values can be determined are called nodal points and they are found at the boundary of the element.

1.2.3 Scope and Limitations of FEM

Various branches of mechanical engineering such as automotive, aeronautical and biomechanical use integrated FEM packages in product design and development. In early days FEM packages were limited to structural simulation producing the structural change and strength visualization of product and helping in optimization of weight, material and cost. Today several FEM packages include the additional fluid, thermal and electromagnetic working environment along with structural indicating the broadened spectrum of FEM. In summary, the major advantages of FEM include- (1)Easier modelling of complex geometrical shapes, (2)Adaptability to meet certain accuracy specifications that strikes out the requirement of physical prototype, (3)Flexibility in applying and changing the boundary conditions like forces and positional constraints and (4)Easily spotting the vulnerability in design with detailed visualizations[23][24].

Given that FEM is based on application of numerical methods for a discrete system, the solution will always be approximate to the theoretical solution obtained by solving the governing equations for a continuous system. As we increase the number of elements, the solution approaches the ideal value but in doing so we also make it computationally more expensive. So, a lot of time (hours or even days) and high end computers with large processing capacity in combination with memory units are required for solving complex engineering problems.

1.3 Machine Learning Approach

The standard FEA approach is great for design and validation of components. But, in present age of Industry 4.0 the standard FEA approach could prove to be slow and lagging when we reach the point of operations or early development and deployment. The highly reliable simulations which are in process during the validation and late design stages could be slow and inadequate to match the quick pace of product operation, daily manufacturing or urgent requirements. This is where we can use the Machine Learning approach of data-driven simulations. The past historical data from the simulation runs can serve as knowledge base for the Machine Learning model. The ML algorithm can then predict the characteristics of new possible product by learning and interpreting the relations between the parameters from the database. The use of Machine Learning approach thus reduces the number of simulation runs during design of new, but similar product[25][14].

To summarize, lets consider a crash scenario of a car door. The crash worthiness of different designs of car door can be represented by data collected from simulation runs of crash scenario for those designs. The Machine Learning model is fit on this data to learn the crash behaviour. Now when testing a new deign, it will be scored by using this trained model instead of FE simulation. So before conducting the computationally expensive simulation, the ML model is able to predict the result and validate the proposed new design[14].

1.4 Scope of Work

This thesis focus on developing an architecture using machine learning techniques, specifically the Deep Neural Networks to mimic the performance of FEA. It does so by making use of graph neural networks that have structural analogy similar to discretization in the FEM method. The case study problem of a cantilever beam is modeled as a graph network to capture the structural information and input force data. The information is then processed through multiple layers in neural network architecture to give final output in form of stress and deformation. The final goal is to train and tune the neural network model to get results similar to the ones from actual FEA software.

The next part of the Thesis is structured as follows:
Chapter 2 provides an insight of the literature and related work in field of Machine Learning and Simulations.
Chapter 3 dives in depth of the actual know-how of the FEM used in FEA.
Chapter 4 gives detailed information about DNNs and how they work.
Chapter 5 describes the case study scenario of cantilever beam in detail and how to develop a framework by integrating FEM and DNN techniques to get the required result.
Chapter 6 contains the results and analysis of various experiments performed on the problem case scenario.
Chapter 7 concludes the thesis by summarizing the results obtained. We also discuss the future possibilities that would expand the horizon of our framework and make it worth for bringing a change in the world of simulations.

Chapter 2

Literature Review

FEA and Machine Learning are important factors driving the Industry 4.0 revolution. While both have similar goals of predicting the behaviour of a system, there has been a considerable amount of research to integrate both in an effort to optimize the performance and obtain better results in lesser amount of time. [26] shows a deep learning framework for simulation in the field of microelectronics. Another paper [27] shows simulation of complex physics problems using graph networks. [28] displays the use of deep learning to accelerate simulations in the field of fluid dynamics. [29] discusses a hybrid approach combining the data-based and knowledge based modelling. It divides this approach into three subfields namely- simulation-assisted machine learning, machine-learning assisted simulation and a hybrid approach with an interplay between both. The paper provides a conceptual framework and through some use cases and methods it displays the versatility, covering simulation-based data augmentation, checking scientific consistency of machine learning models and pattern detection in simulation.

It is not just the simulation part of CAE where machine learning techniques are giving an enhancement to the traditional methods, but also in the design phase of CAD. This paper[30] shows the integration of deep learning techniques into CAD system as well. An actual problem case scenario of design and evaluation of 3D conceptual wheel is used to explain the framework. Their proposed framework consisted of seven stages namely- 2D generated designs, dimensionality reduction, design of experiment in latent space, CAD automation, CAE automation, transfer learning and visualization of results. The initial stages(1-4) in the framework generates 3D wheel CAD data for deep learning by using 2D generative design and 3D automation techniques. The later stages(5-7) of the framework provides a deep learning based model to evaluate the conceptual designs of wheel and give a relation between geometry and engineering performance, thus helping the engineers to make reliable decisions.

The wide range of FEA application also includes the medical domain where it is used to study the biomechanics of human tissues and organs. [31] discusses a deep learning approach as a surrogate of FEA. It states that patient specific FEA models usually require large setup and computation time to get final simulation results. The flow of deep learning model was in following sequence- encoding the input shape of aorta (main artery that carries blood away from the heart to the rest of human body) as a set of scalar values (called as shape code) using PCA, conducting a non-linear mapping of the shape code (input) to the stress code (output) using deep neural network, and finally decoding the stress code into aorta wall stress distribution. The deep learning model was designed and trained on historical data

of aorta shapes and corresponding wall stress distribution to predict the aortic wall stress distribution of a new patient bypassing the FEA calculation process.

Deep learning has already proved its worth in applications like computer vision, natural language processing, audio analysis etc. where the data is in structured grid-like form in Euclidean domain. But there are several other applications where the underlying structure of data is in non-Euclidean space. Some of the examples are- social network in social sciences, molecular data in chemistry, meshed surfaces in computer graphics, sensor networks in communication and so on[32]. A new branch in deep learning known as 'Geometric Deep Learning' implementing the GNN has emerged to deal with such examples. [32] and [6] give a general design pipeline of GNN's and also discuss wide variety of applications domains where they have proved to be highly effective in getting the solution.[32] and [6] also discuss about application of GNN's in the physics domain where a physical system can be modeled as objects and pair wise interaction between them. These objects are called as nodes and the pairwise relations are called as edges. The resulting structure from nodes and edges is called a graph. Thus we can model the physical systems as simplified graphs to learn the law of system and make predictions about the state.

This thesis recognizes the analogy between GNN and FEA in a way that that the structural relational information obtained after the meshing procedure using descretization in FEM could be modeled in form of nodes and edges of a graph in GNN. This analogy can be used to build a Graph Network that would actually represent the physical phenomenon. The next goal would be to train the model in such a way that it learns and interprets various parameters of actual system and gives solution equivalent to the FEA case.

Graph Neural Networks being the recent addition in the Deep Learning umbrella, a lot of research is ongoing to exploit its flexibility. Chapter 3 will dive in deep for us to know the important concepts in Deep Neural Networks.

Chapter 3

Fundamentals of Finite Element Method

The need for solving complex elasticity and structural problems in civil and aeronautical engineering is what laid the foundations of FEM in the field of Finite Element simulations during the 1940's. FEM is a numerical technique used to find the approximate solution of Partial Differential Equations that govern any physical phenomenon. It allows numerical analysis of complex problems under given boundary conditions. Over the years FEM has gained increased popularity in solving complex engineering and science problems [23][33].

3.1 Strong Formulation of Continuous System

The governing Partial Differential Equation along with the boundary conditions for a physical system is known as strong formulation of a continuous system[34]. These equations help in understanding the ideal behaviour of a system. Solving these system of equations one can obtain the theoretical ideal value of unknown output variables at any location in a continuous domain. The analytical solution obtained serves as a benchmark for other methods.

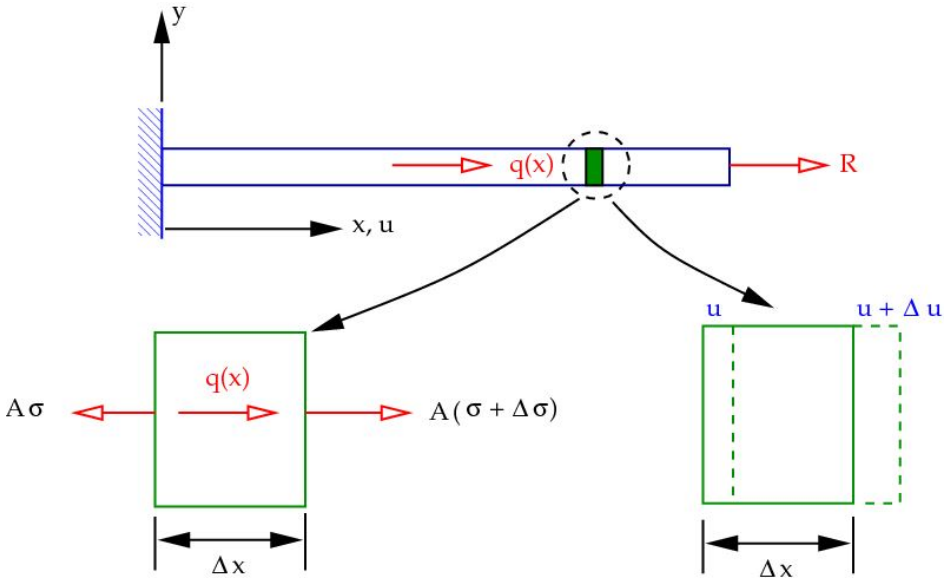

FIGURE 3.1: A continuous system of axially loaded bar [35]

Consider an example of 1D axially loaded bar as shown in figure 3.1. Let's derive the strong formulation of this system to get the insights of the procedure [36]. The bar is fixed at one end and loaded with axial force R at the other. In addition, there is also a line load $q(x)$ acting along the length of bar.

From the condition of equilibrium for an infinitesimal small element of length Δx we have the equation,

$$A\boldsymbol{\sigma} = \mathbf{q}(\mathbf{x})\,\Delta x + A(\boldsymbol{\sigma} + \Delta\boldsymbol{\sigma}) \qquad \Longrightarrow \qquad A\left(\frac{(\boldsymbol{\sigma} + \Delta\boldsymbol{\sigma}) - \boldsymbol{\sigma}}{\Delta x}\right) + \mathbf{q}(\mathbf{x}) = 0$$

$\boldsymbol{\sigma}$ represents stress, while A is the cross section area of the bar. Since Δx is very small we take limit as $\Delta x \to 0$ to get,

$$A \lim_{\Delta x \to 0}\left(\frac{(\boldsymbol{\sigma} + \Delta\boldsymbol{\sigma}) - \boldsymbol{\sigma}}{\Delta x}\right) + \mathbf{q}(\mathbf{x}) = 0 \qquad \Longrightarrow \qquad A\frac{d\boldsymbol{\sigma}}{dx} + \mathbf{q}(\mathbf{x}) = 0$$

This gives the differential equation in terms of stress. Converting it into terms of strain $\boldsymbol{\varepsilon}$ using the relation $\boldsymbol{\sigma} = E\boldsymbol{\varepsilon}$ where E is the Young's Modulus, the equation is modified as,

$$AE\frac{d\boldsymbol{\varepsilon}}{dx} + \mathbf{q}(\mathbf{x}) = 0$$

Expressing the strain term as ratio of change in length to the original length

$$\boldsymbol{\varepsilon} = \lim_{l \to 0}\frac{\Delta l}{l} \qquad \Longrightarrow \qquad \boldsymbol{\varepsilon} = \lim_{\Delta x \to 0}\frac{(\mathbf{u} + \Delta\mathbf{u}) - \mathbf{u}}{\Delta x} \equiv \boldsymbol{\varepsilon} = \frac{d\mathbf{u}}{dx}$$

The differential equation in terms of displacement u is as follows

$$AE\frac{d^2\mathbf{u}}{dx^2} + \mathbf{q}(\mathbf{x}) = 0$$

Considering the line load as a linear function of x we substitute $q(x) = ax$

$$AE\frac{d^2\mathbf{u}}{dx^2} + a\mathbf{x} = 0$$

By solving this ordinary differential equation it is possible to get the displacements in the bar, using which one can also obtain strain and stress values. Note that the higher dimension problems will have partial differential equation in place of ordinary differential equation. To solve the above differential equation, the next step is to substitute the boundary conditions as- at $x = 0$ displacement $u = 0$ and at $x = L$ force $f = R$. Before substituting this conditions force f can be written in terms of displacement as,

$$\mathbf{f} = A\boldsymbol{\sigma} = AE\boldsymbol{\varepsilon} = AE\frac{d\mathbf{u}}{dx} = \mathbf{R}$$

The final differential equation or the strong form can now be written as,

$$\begin{aligned} AE\frac{d^2\mathbf{u}}{dx^2} &= -a\mathbf{x} \\ \mathbf{u}(0) &= 0 \\ \left.\frac{d\mathbf{u}}{dx}\right|_{\mathbf{x}=L} &= \frac{\mathbf{R}}{AE} \end{aligned} \tag{3.1}$$

3.2 Weak Formulation and Finite Element Solution

The weak formulation is basically the integral form of the strong formulation needed to formulate the finite element method[21]. Getting the solution to the strong form can often be difficult specially in case of higher order differentials. In such scenario we need to resort to numerical methods like FEM for obtaining approximate solutions which are favourable for computing. The FEM is applied on a discrete system and converts the partial differential equation to a set of linear algebraic equations[37]. It works on the principle of interpolation of polynomial functions on discrete elements. It first decomposes the continuous complex geometric domain into simpler sub-domains called as elements. The grid obtained by collection of elements is called as finite element mesh. The points on the boundary of elements are called as nodes. This complete process of generating the elements, nodes and the mesh is known as discretization[33]. The different types of elements used in FEM are shown in the figure below.

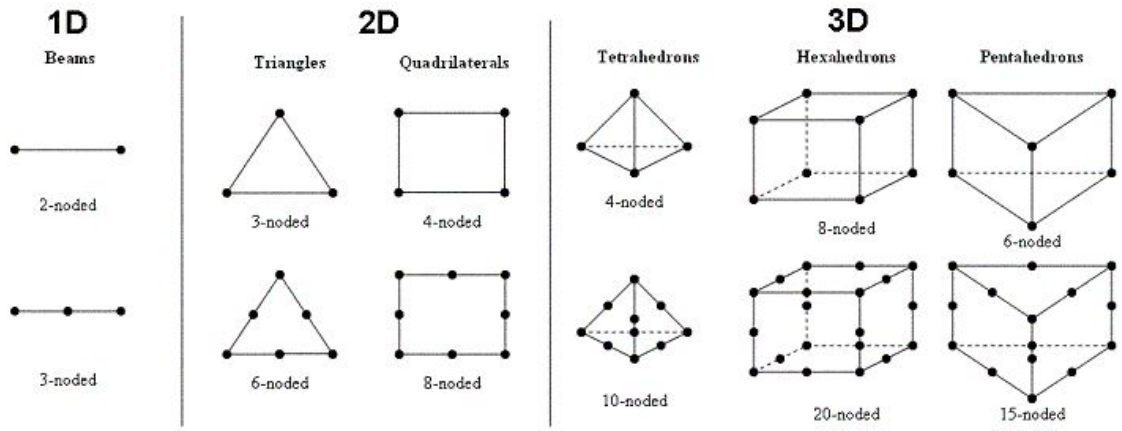

FIGURE 3.2: Different types of finite elements [38]

A common approach of obtaining the finite element solution is through Galerkin method of weighted residual. Before that we need to prepare the weak formulation [21][39]. Lets consider approximate solution $u_h(x)$ to transform the strong form

differential equation into,

$$AE\frac{d^2\mathbf{u}_h}{dx^2} + a\mathbf{x} = R_h(\mathbf{x}) \tag{3.2}$$

R_h is the weighted average error or residual which we aim to minimize with weighting function $w(x)$ as

$$\int_0^L R_h(\mathbf{x})\mathbf{w}(\mathbf{x})\, dx = 0$$

Plugging in the weighting function, the approximate weak form can be written as,

$$\int_0^L AE\frac{d\mathbf{u}_h}{dx}\frac{d\mathbf{w}}{dx}\, dx = \int_0^L \mathbf{qw}\, dx + \boldsymbol{R}\ \mathbf{w}|_{x=L}$$

The approximate solution in Galerkin method is written as follows.

$$\mathbf{u}_h(\mathbf{x}) = a_1\varphi_1(\mathbf{x}) + a_2\varphi_2(\mathbf{x}) + \cdots + a_n\varphi_n(\mathbf{x}) = \sum_{i=1}^{n} a_i\varphi_i(\mathbf{x})$$

The weighting function is also of the same form with arbitrary coefficients

$$\mathbf{w}(\mathbf{x}) = b_1\varphi_1(\mathbf{x}) + b_2\varphi_2(\mathbf{x}) + \cdots + b_n\varphi_n(\mathbf{x}) = \sum_{j=1}^{n} b_j\varphi_j(\mathbf{x})$$

Substituting the approximate solution and weighting function gives,

$$\int_0^L AE\left(\sum_{i=1}^{n} a_i\frac{d\varphi_i}{dx}\right)\left(\sum_{j=1}^{n} b_j\frac{d\varphi_j}{dx}\right)\, dx = \int_0^L \mathbf{q}\left(\sum_{j=1}^{n} b_j\varphi_j\right)\, dx + \boldsymbol{R}\ \left(\sum_{j=1}^{n} b_j\varphi_j\right)\Bigg|_{x=L}$$

Modifying the equation as b_j is arbitrary

$$\sum_{j=1}^{n} b_j\left[\int_0^L AE\left(\sum_{i=1}^{n} a_i\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}\right)\, dx\right] = \sum_{j=1}^{n} b_j\left[\int_0^L \mathbf{q}\varphi_j\, dx + \left(\boldsymbol{R}\ \varphi_j\right)\big|_{x=L}\right] \tag{3.3}$$

$$\tag{3.4}$$

$$\int_0^L AE\left(\sum_{i=1}^{n} a_i\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}\right)\, dx = \int_0^L \mathbf{q}\varphi_j\, dx + \boldsymbol{R}\ \varphi_j\big|_{x=L}\ ,\ j = 1\ldots n. \tag{3.5}$$

$$\tag{3.6}$$

$$\sum_{i=1}^{n}\left[\int_0^L \frac{d\varphi_j}{dx}AE\frac{d\varphi_i}{dx}\, dx\right]a_i = \int_0^L \varphi_j\mathbf{q}\, dx + \varphi_j\boldsymbol{R}\big|_{x=L}\ ,\ j = 1\ldots n \tag{3.7}$$

The equation 3.7 is the final weak formulation representing a system of n equations to be solved for coefficients a_i which are unknown. The approximate solution can

be obtained given we have values of a_i. The matrix form of the equation in terms of stiffness matrix K and load matrix f can be written as,

$$\mathbf{K\alpha} = \mathbf{f} \qquad \leftrightarrow \qquad K_{ji}a_i = f_j \tag{3.8}$$

After obtaining the weak formulation, the next step is to proceed towards the finite element solution using Galerkin method. In Galerkin method, the function ψ_i also represented as N_i are called as shape functions or interpolation functions and have the following properties: (1)they are systematic , (2)they can be chosen and used for different random domains, (3)they represent piece-wise polynomials, (4)they have non-zero value over limited domain

Consider the earlier example of 1D axially loaded bar. While previously the analytical solution was obtained by means of differential equation for the continuous system, this time the finite element approach will be used to obtain the approximate solution[40][21]. The first step is to generate finite element mesh and to initialize the shape functions over finite domains as shown in figure 3.3.

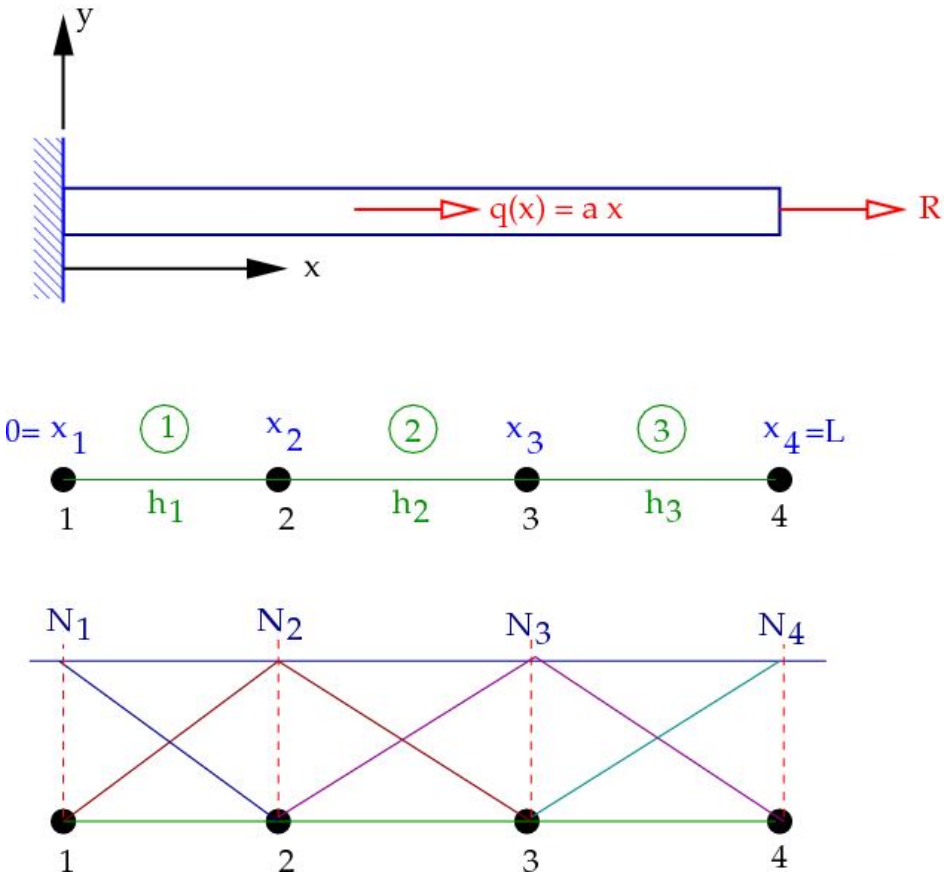

FIGURE 3.3: Finite elements and shape functions for axially loaded bar [35]

The weak formulation in terms of shape function N can be written as

$$\begin{aligned} K_{ij} &= \int_0^L \frac{dN_j}{dx} AE \frac{dN_i}{dx}\, dx \\ f_j &= \int_0^L N_j \mathbf{q}\, dx + N_j \boldsymbol{R}\big|_{x=L} \end{aligned} \tag{3.9}$$

Every shape function has value 1 at their respective node. The stiffness values K_{ij} for mesh with three elements can be computed using the formula,

$$K_{ij} = \int_0^L \frac{dN_j}{dx} AE \frac{dN_i}{dx}\, dx$$

$$= \int_0^{x_2} \frac{dN_j}{dx} AE \frac{dN_i}{dx}\, dx + \int_{x_2}^{x_3} \frac{dN_j}{dx} AE \frac{dN_i}{dx}\, dx + \int_{x_3}^{L} \frac{dN_j}{dx} AE \frac{dN_i}{dx}\, dx$$

$$= \int_{\Omega_1} \frac{dN_j}{dx} AE \frac{dN_i}{dx}\, dx + \int_{\Omega_2} \frac{dN_j}{dx} AE \frac{dN_i}{dx}\, dx + \int_{\Omega_3} \frac{dN_j}{dx} AE \frac{dN_i}{dx}\, dx$$

As there are four nodes, the stiffness matrix will be a 4x4 matrix with values from K_{11} to K_{44}. In the figure 3.3, it can be seen that the shape function value is non-zero for a specific domain only. Substituting the zero value for N_1 in element 2 and 3, N_2 in element 3, N_3 in element 1, and N_4 in element 1 and 2, gives the coefficients of stiffness matrix as,

$$K_{11} = \int_{\Omega_1} \frac{dN_1}{dx} AE \frac{dN_1}{dx}\, dx \;;\; K_{12} = \int_{\Omega_1} \frac{dN_2}{dx} AE \frac{dN_1}{dx}\, dx \;;\; K_{13} = 0 \;;\; K_{14} = 0$$

$$K_{22} = \int_{\Omega_1} \frac{dN_2}{dx} AE \frac{dN_2}{dx}\, dx + \int_{\Omega_2} \frac{dN_2}{dx} AE \frac{dN_2}{dx}\, dx$$

$$K_{23} = \int_{\Omega_2} \frac{dN_3}{dx} AE \frac{dN_2}{dx}\, dx \;;\; K_{24} = 0$$

$$K_{33} = \int_{\Omega_2} \frac{dN_3}{dx} AE \frac{dN_3}{dx}\, dx + \int_{\Omega_3} \frac{dN_3}{dx} AE \frac{dN_3}{dx}\, dx \;;\; K_{34} = \int_{\Omega_3} \frac{dN_4}{dx} AE \frac{dN_3}{dx}\, dx$$

$$K_{44} = \int_{\Omega_3} \frac{dN_4}{dx} AE \frac{dN_4}{dx}\, dx$$

As the K matrix is symmetric there is no need to calculate the other remaining terms separately. The next step is to modify the representation for shape function N_i as N_k^e representing the shape function over element e. For example, writing the shape functions over element 2 as N_1^2 and N_2^2 where local nodes 1 and 2 are global nodes 2 and 3 respectively.

$$K_{11} = \int_{\Omega_1} \frac{dN_1^1}{dx} AE \frac{dN_1^1}{dx}\, dx \; ; \; K_{12} = \int_{\Omega_1} \frac{dN_2^1}{dx} AE \frac{dN_1^1}{dx}\, dx \; ; \; K_{13} = 0 \; ; \; K_{14} = 0$$

$$K_{22} = \int_{\Omega_1} \frac{dN_2^1}{dx} AE \frac{dN_2^1}{dx}\, dx + \int_{\Omega_2} \frac{dN_1^2}{dx} AE \frac{dN_1^2}{dx}\, dx$$

$$K_{23} = \int_{\Omega_2} \frac{dN_2^2}{dx} AE \frac{dN_1^2}{dx}\, dx \; ; \; K_{24} = 0$$

$$K_{33} = \int_{\Omega_2} \frac{dN_2^2}{dx} AE \frac{dN_2^2}{dx}\, dx + \int_{\Omega_3} \frac{dN_1^3}{dx} AE \frac{dN_1^3}{dx}\, dx \; ; \; K_{34} = \int_{\Omega_3} \frac{dN_2^3}{dx} AE \frac{dN_1^3}{dx}\, dx$$

$$K_{44} = \int_{\Omega_3} \frac{dN_2^3}{dx} AE \frac{dN_2^3}{dx}\, dx$$

Let K_{kl}^e be the contribution of element e in K_{ij} where kl are local indices and ij are global indices.

$$\begin{aligned}
&K_{11} = K_{11}^1 \; ; \; K_{12} = K_{12}^1 \; ; \; K_{13} = 0 \; ; \; K_{14} = 0 \\
&K_{22} = K_{22}^1 + K_{11}^2 \; ; \; K_{23} = K_{12}^2 \; ; \; K_{24} = 0 \\
&K_{33} = K_{22}^2 + K_{11}^3 \; ; \; K_{34} = K_{12}^3 \\
&K_{44} = K_{22}^3
\end{aligned}$$

Using these equations one can compute the stiffness matrices over each element and assemble them to get the global stiffness matrix K.

For an element e with length h_e between two nodes, the local hat shape functions are of the form,

$$N_1^e(\mathbf{x}) = \frac{\mathbf{x}_2 - \mathbf{x}}{h_e} \; ; \; N_2^e(\mathbf{x}) = \frac{\mathbf{x} - \mathbf{x}_1}{h_e}$$

The element stiffness matrix will have following components,

$$K_{11}^e = \int_{x_1}^{x_2} \frac{dN_1^e}{dx} AE \frac{dN_1^e}{dx}\, dx = \int_{x_1}^{x_2} \left(-\frac{1}{h_e}\right) AE \left(-\frac{1}{h_e}\right) dx = \frac{AE}{h_e}$$

$$K_{12}^e = \int_{x_1}^{x_2} \frac{dN_2^e}{dx} AE \frac{dN_1^e}{dx}\, dx = \int_{x_1}^{x_2} \left(\frac{1}{h_e}\right) AE \left(-\frac{1}{h_e}\right) dx = -\frac{AE}{h_e}$$

$$K_{22}^e = \int_{x_1}^{x_2} \frac{dN_2^e}{dx} AE \frac{dN_2^e}{dx}\, dx = \int_{x_1}^{x_2} \left(\frac{1}{h_e}\right) AE \left(\frac{1}{h_e}\right) dx = \frac{AE}{h_e}$$

Substituting the values in respective position in matrix,

$$\mathbf{K}^e = \frac{AE}{h_e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Using the matrix form we get the global stiffness matrix components as,

$$\begin{aligned}
K_{11} &= \frac{AE}{h_1}\,;\ K_{12} = -\frac{AE}{h_1}\,;\ K_{13} = 0\,;\ K_{14} = 0 \\
K_{22} &= \frac{AE}{h_1} + \frac{AE}{h_2}\,;\ K_{23} = -\frac{AE}{h_2}\,;\ K_{24} = 0 \\
K_{33} &= \frac{AE}{h_2} + \frac{AE}{h_3}\,;\ K_{34} = -\frac{AE}{h_3} \\
K_{44} &= \frac{AE}{h_3}
\end{aligned}$$

Assembling the stiffness matrix components gives the final global stiffness matrix as shown in equation 3.10

$$\mathbf{K} = AE \begin{bmatrix} \frac{1}{h_1} & -\frac{1}{h_1} & 0 & 0 \\ & \frac{1}{h_1} + \frac{1}{h_2} & -\frac{1}{h_2} & 0 \\ & & \frac{1}{h_2} + \frac{1}{h_3} & -\frac{1}{h_3} \\ \text{Symm.} & & & \frac{1}{h_3} \end{bmatrix} \tag{3.10}$$

The next step is to compute the load vector. The load vector equation is of the form,

$$\begin{aligned} f_j &= \int_0^L N_j \mathbf{q}\, dx + N_j \boldsymbol{R}\big|_{x=L} \\ &= \int_{\Omega_1} N_j \mathbf{q}\, dx + \int_{\Omega_2} N_j \mathbf{q}\, dx + \int_{\Omega_3} N_j \mathbf{q}\, dx + N_j \boldsymbol{R}\big|_{x=L} \end{aligned}$$

Substituting the zero values for shape function over elements where they don't exist, and also substituting the boundary condition of force R for N_4 gives,

$$\begin{aligned} f_1 &= \int_{\Omega_1} N_1 \mathbf{q}\, dx \\ f_2 &= \int_{\Omega_1} N_2 \mathbf{q}\, dx + \int_{\Omega_2} N_2 \mathbf{q}\, dx \\ f_3 &= \int_{\Omega_2} N_3 \mathbf{q}\, dx + \int_{\Omega_3} N_3 \mathbf{q}\, dx \\ f_4 &= \int_{\Omega_3} N_4 \mathbf{q}\, dx + N_4 \boldsymbol{R}|_{x=L} \end{aligned}$$

Modifying the equations in terms of element shape function,

$$\begin{aligned} f_1 &= \int_{\Omega_1} N_1^1 \mathbf{q}\, dx = f_1^1 \\ f_2 &= \int_{\Omega_1} N_2^1 \mathbf{q}\, dx + \int_{\Omega_2} N_1^2 \mathbf{q}\, dx = f_2^1 + f_1^2 \\ f_3 &= \int_{\Omega_2} N_2^2 \mathbf{q}\, dx + \int_{\Omega_3} N_1^3 \mathbf{q}\, dx = f_2^2 + f_1^3 \\ f_4 &= \int_{\Omega_3} N_2^3 \mathbf{q}\, dx + N_2^3 \boldsymbol{R}\Big|_{x=L} = f_2^3 + \boldsymbol{R} \end{aligned}$$

Using the element notation to get the element load vector f^e and replacing q with ax gives,

$$\begin{aligned} f_1^e &= \int_{x_1}^{x_2} N_1^e a\mathbf{x}\, dx = \int_{x_1}^{x_2} \left(\frac{x_2 - x}{h_e}\right) a\mathbf{x}\, dx = \frac{a}{h_e}\left(\frac{x_2(x_2^2 - x_1^2)}{2} - \frac{x_2^3 - x_1^3}{3}\right) \\ f_2^e &= \int_{x_1}^{x_2} N_2^e a\mathbf{x}\, dx = \int_{x_1}^{x_2} \left(\frac{x - x_1}{h_e}\right) a\mathbf{x}\, dx = -\frac{a}{h_e}\left(\frac{x_1(x_2^2 - x_1^2)}{2} - \frac{x_2^3 - x_1^3}{3}\right) \end{aligned}$$

The matrix form can be written as,

$$\mathbf{f}^e = \frac{a}{h_e}\begin{bmatrix} \frac{x_2(x_2^2-x_1^2)}{2} - \frac{x_2^3-x_1^3}{3} \\ \frac{x_2^3-x_1^3}{3} - \frac{x_1(x_2^2-x_1^2)}{2} \end{bmatrix} \tag{3.11}$$

The components of global load vector are as follows,

$$\begin{aligned}
f_1 &= \frac{a}{h_1}\left(\frac{x_2(x_2^2-x_1^2)}{2} - \frac{x_2^3-x_1^3}{3}\right) \\
f_2 &= \frac{a}{h_1}\left(\frac{x_2^3-x_1^3}{3} - \frac{x_1(x_2^2-x_1^2)}{2}\right) + \frac{a}{h_2}\left(\frac{x_3(x_3^2-x_2^2)}{2} - \frac{x_3^3-x_2^3}{3}\right) \\
f_3 &= \frac{a}{h_2}\left(\frac{x_3^3-x_2^3}{3} - \frac{x_2(x_3^2-x_2^2)}{2}\right) + \frac{a}{h_3}\left(\frac{x_4(x_4^2-x_3^2)}{2} - \frac{x_4^3-x_3^3}{3}\right) \\
f_4 &= \frac{a}{h_3}\left(\frac{x_4^3-x_3^3}{3} - \frac{x_3(x_4^2-x_3^2)}{2}\right) + \boldsymbol{R}
\end{aligned}$$

The approximate solution in form of displacement function defined in the weak formulation part can now be modified as,

$$\mathbf{u}_h(\mathbf{x}) = a_1\varphi_1(\mathbf{x}) + a_2\varphi_2(\mathbf{x}) + \cdots + a_n\varphi_n(\mathbf{x}) = \sum_{i=1}^{n} a_i\varphi_i(\mathbf{x})$$

$$\mathbf{u}_h(\mathbf{x}) = a_1\, N_1(\mathbf{x}) + a_2\, N_2(\mathbf{x}) + \cdots + a_n\, N_n(\mathbf{x}) = \sum_{i=1}^{n} a_i\, N_i(\mathbf{x})$$

Substituting $N_i = 1$ at respective node and zero elsewhere gives,

$$\begin{aligned}
u_1 &:= \mathbf{u}_h(\mathbf{x}_1) = a_1\, N_1(\mathbf{x}_1) + a_2\, N_2(\mathbf{x}_1) + \cdots + a_n\, N_n(\mathbf{x}_1) = a_1 \\
u_2 &:= \mathbf{u}_h(\mathbf{x}_2) = a_1\, N_1(\mathbf{x}_2) + a_2\, N_2(\mathbf{x}_2) + \cdots + a_n\, N_n(\mathbf{x}_2) = a_2 \\
&\vdots \\
u_n &:= \mathbf{u}_h(\mathbf{x}_n) = a_1\, N_1(\mathbf{x}_n) + a_2\, N_2(\mathbf{x}_n) + \cdots + a_n\, N_n(\mathbf{x}_n) = a_n
\end{aligned}$$

The trial function in terms of nodal displacements u_i can be written as

$$\mathbf{u}_h(\mathbf{x}) = u_1\, N_1(\mathbf{x}) + u_2\, N_2(\mathbf{x}) + \cdots + u_n\, N_n(\mathbf{x}) = \sum_{i=1}^{n} u_i\, N_i(\mathbf{x}) \tag{3.12}$$

Assuming all elements have same length h the finite element system of equations $Ka = f$ is of the form,

$$\frac{AE}{h}\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

$$= \frac{a}{h}\begin{bmatrix} \left(\frac{x_2(x_2^2-x_1^2)}{2} - \frac{x_2^3-x_1^3}{3}\right) \\ \left(\frac{x_2^3-x_1^3}{3} - \frac{x_1(x_2^2-x_1^2)}{2}\right) + \left(\frac{x_3(x_3^2-x_2^2)}{2} - \frac{x_3^3-x_2^3}{3}\right) \\ \left(\frac{x_3^3-x_2^3}{3} - \frac{x_2(x_3^2-x_2^2)}{2}\right) + \left(\frac{x_4(x_4^2-x_3^2)}{2} - \frac{x_4^3-x_3^3}{3}\right) \\ \left(\frac{x_4^3-x_3^3}{3} - \frac{x_3(x_4^2-x_3^2)}{2}\right) + \boldsymbol{R}\frac{h}{a} \end{bmatrix} \tag{3.13}$$

Substituting the boundary condition $u = 0$ at $x = 0$ we simplify the equation as,

$$\frac{AE}{h}\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \frac{a}{h}\begin{bmatrix} \left(\frac{x_2^3-x_1^3}{3} - \frac{x_1(x_2^2-x_1^2)}{2}\right) + \left(\frac{x_3(x_3^2-x_2^2)}{2} - \frac{x_3^3-x_2^3}{3}\right) \\ \left(\frac{x_3^3-x_2^3}{3} - \frac{x_2(x_3^2-x_2^2)}{2}\right) + \left(\frac{x_4(x_4^2-x_3^2)}{2} - \frac{x_4^3-x_3^3}{3}\right) \\ \left(\frac{x_4^3-x_3^3}{3} - \frac{x_3(x_4^2-x_3^2)}{2}\right) + \boldsymbol{R}\frac{h}{a} \end{bmatrix} \tag{3.14}$$

Given the values of A, E, L and R it is possible to solve this final system of equation to get displacements u_2, u_3 and u_4. The displacement values can then be used to calculate strains and stresses. This completes the FEM method of obtaining approximate solution.

To summarize, the general methodology is converting the strong form which is the PDE governing the physical problem to a some weak formulation to give approximate solution over elements. One of the method to obtain the weak formulation is by multiplying the PDE with some weight function and test function and expressing it in integral form to integrate it over the domain for which it holds true. This reduces the complexity of the original PDE. Further in Galerkin method, the weight function and test function in the weak form are written as shape function. The next step is to use different shape functions over different domains in finite element mesh. The element stiffness and the nodal deformations are then expressed using the shape functions to get the finite element system of equations with global stiffness matrix. The system of equations is further simplified by substituting the boundary conditions and the result is obtained in form of nodal displacements. Using these values it is possible to further calculate strains and stress.

Chapter 4

Deep Learning

Deep Learning along with neural networks have been around with us since 1943 when a computer model was created based on neural networks of the human brain [41]. But it was from 1980's after numerous paper publications that the neural networks caught the wind and were believed to be something that would change the world. Unfortunately this trend dried out over the next years because of one main reason- the technology at that time was insufficient to facilitate the application of neural networks and deep learning. Neural networks required a lot of data and a strong processing power to process that data which was simply not available during the early days. Later on, the exponential growth in the technology of data storage and computing power made it possible for us to enter the era of computers that are extremely powerful and can process things way faster than we could imagine. This is what is facilitating deep learning today[42].

The human brain is one of the most powerful tool when it comes to learning skills, applying them and adapting to them. The whole idea behind deep learning is to mimic the behaviour of human brain and make computers perform different tasks by leveraging the algorithms adapting the human brain functionality. The last decade has witness success in doing so, as today deep learning is showing excellent results in applications of image processing, natural language processing, signal processing, pattern recognition and so on. Deep Learning has thus emerged as a branch of machine learning that exploits multiple layers of non-linear information processing for supervised or unsupervised feature extraction, transformation, pattern analysis and classification[43]. Some of the popular deep learning algorithms include Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory Networks (LSTM), Graph Neural Networks (GNN), Stacked Auto-Encoders and so on. The further sections of this chapter will dive deep into deep neural networks.

4.1 Artificial Neural Networks

The human brain consists of billions of neurons that are interconnected with each other. They are responsible for taking in the input signals, processing it and giving the required output. A similar artificial structure is recreated for computers where the input nodes or neurons in input layer are followed by multiple hidden layers of neurons and finally output neuron layer in the end. The neurons in the subsequent layers are interconnected to each other. An artificial neural network is thus a computation model inspired by the structure of neural networks in human brain that

uses a network of functions to understand and translate the input data into desired output[44][45]. After getting an overview of artificial neural network, the following subsections will give detailed information of the building blocks of artificial neural networks.

4.1.1 Neuron

A neuron is the main basic building block of an artificial neural network. The figure below shows the analogy between the structure of a biological neuron and a basic neuron unit in a neural network also known as a perceptron. The perceptron was invented in 1957 by Frank Rosenblatt[46].

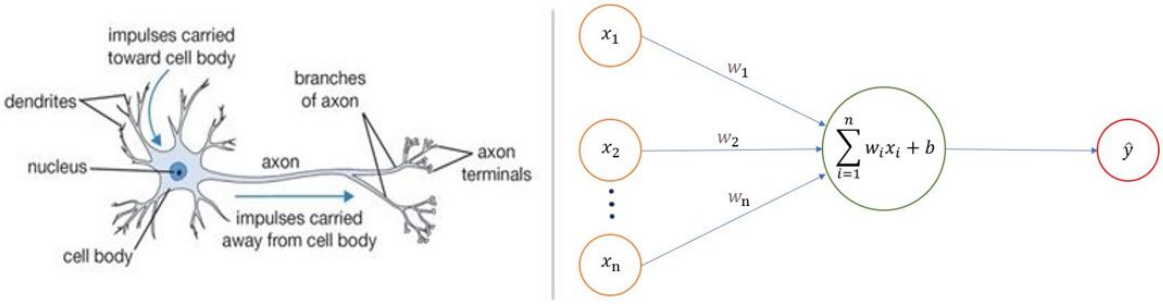

FIGURE 4.1: Biological neuron vs. a perceptron [3]

A single neuron can take in multiple inputs, process them and give an output. The output from a neuron is the weighted sum of all the input values in addition to a bias value. The bias or offset parameter is added to provide a mechanism to include other influences or factors apart from the input data [3].

$$\hat{y} = \sum_{i=1}^{n} w_i x_i + b \tag{4.1}$$

In equation 4.1, $\hat{y}$ is the output, x_i are the inputs, w_i are the weights and b is the bias. The weights along the edges or connections represent the importance of a particular input value and decides to what extent it should be passed further to influence the output.

4.1.2 Activation Function

As seen in previous subsection 4.1.1, the output of a single perceptron is a linear combination of weighted sum of inputs and bias. A neural network is combination of many such basic units and it is essential to introduce some non-linearity to solve complex problems where the relation between input and output is non-linear in nature. Hence a function known as activation function is applied on the output of the neuron[42].

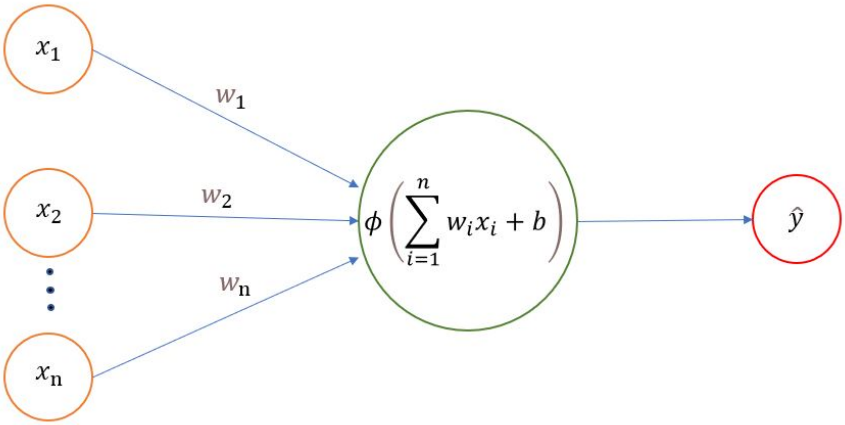

FIGURE 4.2: Perceptron with activation function

The output now obtained is as follows:

$$\hat{y} = \phi(\sum_{i=1}^{n} w_i x_i + b) \tag{4.2}$$

There are few conditions for selecting an activation function. It should be differentiable so that the gradient error backpropagates and it should be non-linear to map the non-linear relation between input and output[44]. Selection of a proper activation function has high impact on the networks ability to learn. Some of the popular activation functions are shown below.

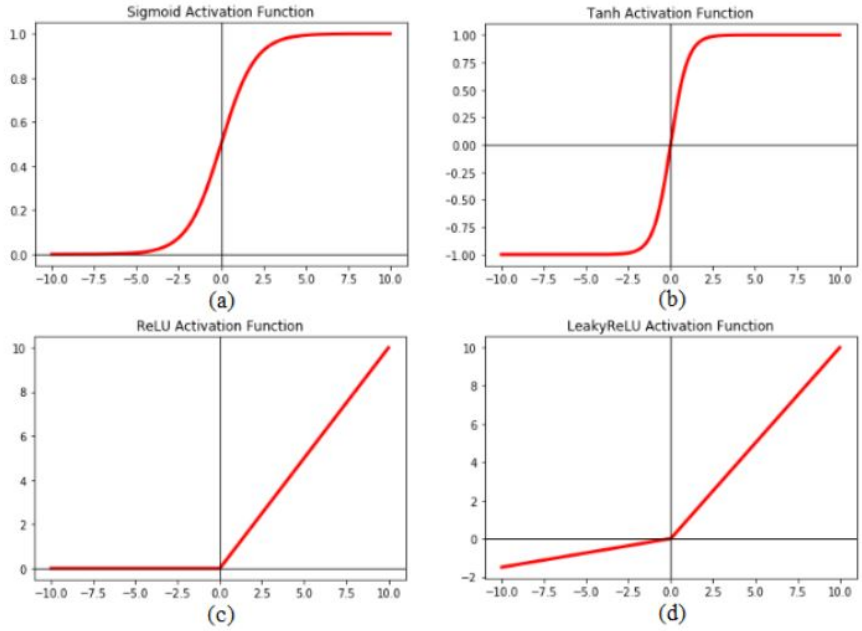

FIGURE 4.3: Different activation functions[4]

In general it is common practice to use the Rectified Linear Unit (ReLU) activation function for the neurons in the hidden layer and a sigmoid activation function

for the output layer in case of a classification task. Regression models generally do not have any activation function for output layer neurons. In this thesis we would be using ReLU and Leaky ReLU activation functions[42].

4.1.3 Working of an Artificial Neural Network

As discussed earlier, a neural network consists of an input layer, multiple hidden layers and an output layer. When an artificial neural network architecture has a single hidden layer, the concept of learning is called as shallow learning where the output is dependent on direct combination of input data. Whereas, when there are multiple hidden layers, then the artificial neural network is called as deep neural network[47][48].

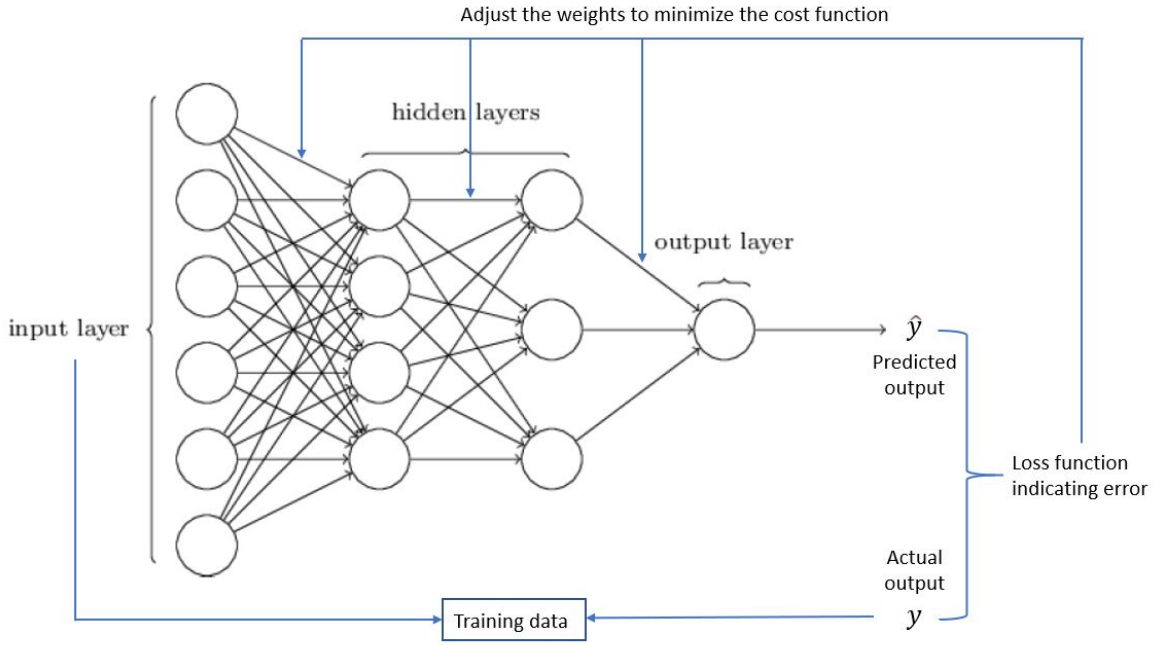

FIGURE 4.4: Basic working principle of ANN[4]

The final output obtained is result of propagation of input data through series of hidden layers. The neurons in each of the hidden layer calculate the weighted sum of inputs, add the bias and pass the result through an activation function. The final output is thus a combination of series of contributions from neurons of every layer. The flow of information from input layer followed by multiple hidden layers and finally the output layer to generate the predicted output is known as forward propagation. The final goal is to get the predicted value as close to the actual value as possible. A function known as cost function is used to indicate the error between actual and predicted value. Some of the examples of cost function are Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), SmoothL1, Binary Cross Entropy etc. The error is propagated backward and the neural network then adjusts the weights and biases from every layer in order to minimize the cost function. This

process of forward propagation and backpropagation continues for multiple iterations known as epochs on a dataset and the network keeps on learning during this process[42].

4.1.4 Gradient Descent

During the backpropagation, the weights are adjusted depending on the cost function value. But how does the neural network actually update the weight? It does so by the method of gradient descent. It is clear that the predicted output depends on the weights used in different layers and so the cost function itself is a function of combination of these weights. One of the method of optimizing the cost function is by brute forcing various combination of weights and looking at which one gives the best result. But this method is prone to curse of dimensionality when we have multiple hidden layers and vast combinations of weights. Gradient descent is a faster and better way of finding the best combination of weights for optimized cost function[5][49]. Consider an example of cost function curve shown in figure 4.5 for different combination of weights.

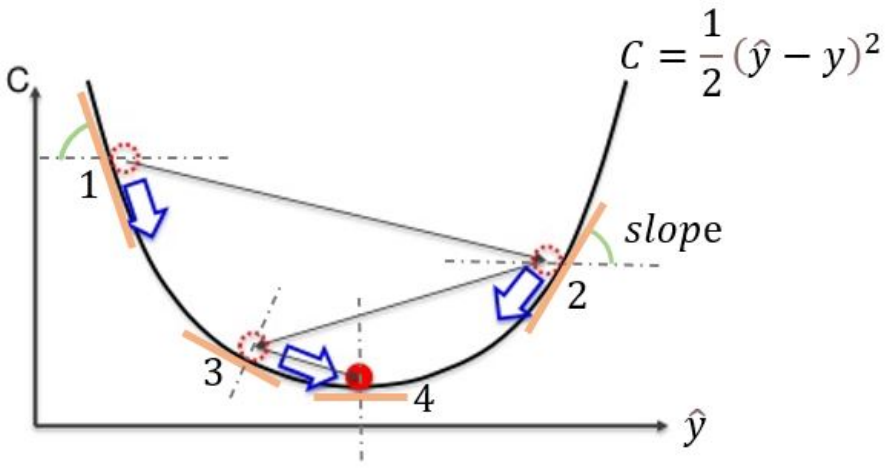

FIGURE 4.5: Gradient Descent method[5]

The aim is to optimize the cost function to give the minimum value for particular combination of weights. Consider random initialization of weights giving the cost function value corresponding to point 1 in the figure. Differentiating the function at point 1 gives the slope at that point also known as gradient of cost function. The gradient thus helps in scaling and changing the value of variables that go as input in cost function (weights and biases) in a direction that would minimize the cost function. For example in the figure after changing the weights, cost function value reaches point 2, later to point 3 and finally at point 4. The gradient at point 4 is zero which indicates a stationary point. There can be more than one stationary points, but the one which has the lowest value of cost function is known as the global minimum[5][49].

4.1.5 Stochastic Gradient Descent

The normal gradient descent also known as vanilla gradient descent or batch gradient descent sums up the cost function value from all the observation rows of the dataset and then backpropagates it update the weight. The entire set of observation is considered as a batch and hence the name- batch gradient descent. The previous example considered a cost function that was convex in nature that had just one local minima which was also the global minima. But what if the cost function we select is not convex? In that case our normal gradient descent could give the output as local minimum instead of global and it would then result in having a sub-par neural network[5].

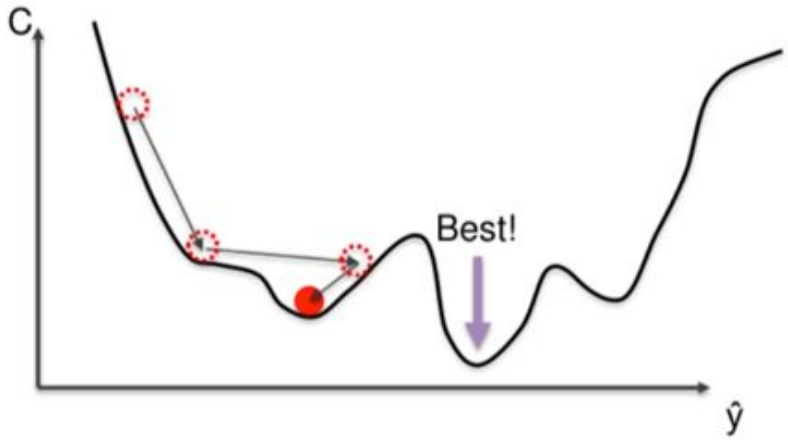

FIGURE 4.6: Batch gradient descent giving local minimum[5]

To solve this problem we have the stochastic gradient descent method. In stochastic gradient descent, unlike normal gradient descent in which the updation of weights takes place after going through the entire batch, the weights are updated after every single row of observation in the dataset. This introduces stochasticity or randomness while learning and avoids getting stuck on a local minimum. Another advantage of stochastic gradient descent over batch gradient descent is that it is much lighter and faster algorithm as it computes for single row at a time and avoids loading of all the data at once. Another method of gradient descent in between the batch and stochastic is the mini-batch gradient descent in which the weights are updated after certain number of observations known as mini-batch. There are various other gradient based learning algorithms like RMSprop, Adagrad, Adam and so on. In this thesis, experimenting will be done with these algorithms to see the performance of the neural net model. Adam is a type of Stochastic gradient descent algorithm that allows faster convergence and adaptive learning rate simultaneously[49].

4.1.6 Backpropagation

Backpropagation is an advanced algorithm driven by sophisticated mathematics that enables the neural network to adjust the weights. Using backpropagation enables us to know for which part of error, each of the weights in neural network is responsible for. In other words, backpropagation repeatedly adjusts the weights of connections simultaneously in the network to minimize the difference between the output of the net and the actual desired output. The gradient of cost function with respect to the parameters or weight determines the level of adjustment of weights[50]. For a cost function C depending on variables $x_1, x_2, \ldots, x_m$ the gradient function is of the form

$$\frac{\partial C}{\partial x} = [\frac{\partial C}{\partial x_1}, \frac{\partial C}{\partial x_2}, \ldots, \frac{\partial C}{\partial x_m}] \tag{4.3}$$

The derivative of cost function gives the sensitivity of change of function value with respect to the parameters which implies the information of how much a parameter needs to change in order to minimize C. Computing of these gradients is done by chain rule[50]. For a single weight w_{jk} in layer l with m neurons in $(l-1)$ layer, the gradient is calculated as follows,

$$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j}\frac{\partial z^l_j}{\partial w^l_{jk}} \tag{4.4}$$

$$z^l_j = \sum_{k=1}^{m} w^l_{jk} a^{l-1}_k + b^l_j \tag{4.5}$$

$$\frac{\partial z^l_j}{\partial w^l_{jk}} = a^{l-1}_k \tag{4.6}$$

$$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial C}{\partial z^l_j} a^{l-1}_k \tag{4.7}$$

Now after knowing the various building block of ANN, let's summarize the training process of ANN in a basic form with the help of following steps:

- Random initialization of weights with small value close to zero.
- Input each feature of the first observation in dataset as node input.
- Forward Propagation- Propagate the activations of each neurons from left to right and get predicted result y. The impact of each neuron is limited by its weight.
- Measure the error by comparing the predicted result with actual result.
- Backpropagation- The error is propagated from right to left. Update the weights based on gradient value that determines how much they are responsible for the error. The learning rate parameter decides by how much we update the weights.

- Repeat the above steps for each observation (stochastic) or for a batch of observations (batch learning).
- After the whole training set is passed through ANN (one epoch), run multiple epochs.

4.2 Graph Neural Networks

4.2.1 What is a Graph?

A graph is a data structure consisting of nodes (vertices) and edges connected together [51]. A graph structure thus models a set of objects (nodes) and their relationships (edges) [6]. A Graph G can be represented as $G = (V, E)$ where V denotes set of nodes and E are the edges between them. The nodes take arbitrary position in space with similar nodes clustered close in 2D or n-dimensional space. The arrows on the edges represent the kind of relationship (two-sided or one-sided) that decides the direction of information flow.

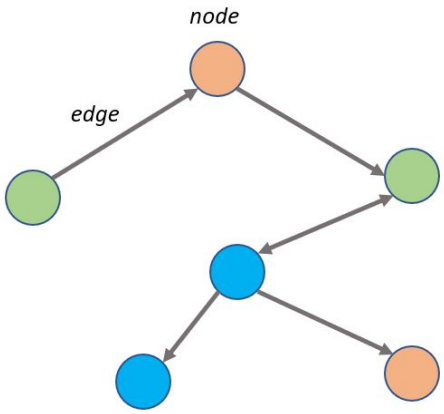

FIGURE 4.7: A Graph

According to [6] graphs can be distinguished as-

- Directed and Undirected graphs: Directed graphs have directional dependencies between nodes (connection direction matters) while undirected graphs do not have such dependencies (connection order doesn't matter). A good example of directed graph would be Twitter social network while Facebook social network is an undirected graph.
- Homogeneous and Heterogeneous graphs: If all the nodes represent instances of same type and all the edges represent relations of same type then it is a homogeneous graph. A heterogeneous graph has multi-type nodes and edges.

Recent researches of analysing graph using machine learning have been gaining a lot of attention because of high expressive power of graphs. Graphs can be used to model large number of systems across various application areas such as social science, physics, chemistry, recommendation systems, knowledge graph and so on[6].

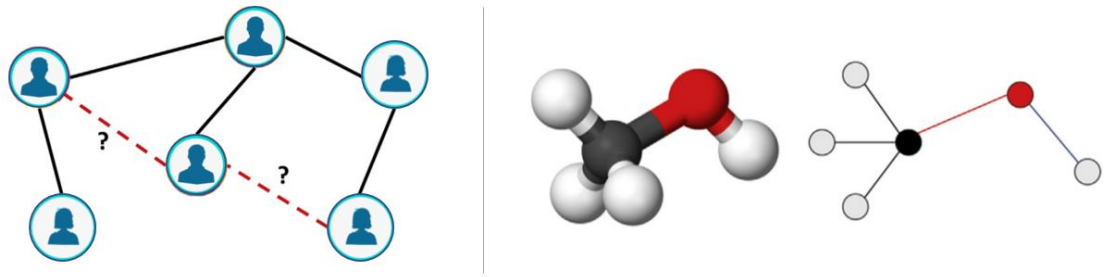

FIGURE 4.8: Social network and a molecule as a graph[6]

Machine learning, especially deep learning methods are good at capturing the patterns on Euclidean data like images, texts, videos where the data is in structured fixed size grid like arrangement of in form of a linear sequence. Graph data on the other hand is from non-Euclidean domains where the shape is arbitrary with no spatial locality or fixed ordering of nodes. Hence it is difficult to perform traditional deep learning methods like CNN on graphs having complex topology and varying size or node ordering. Therefore there is a different branch of deep learning known as geometric deep learning which deals with graphs in form of Graph Neural Networks[52][6][7][53].

4.2.2 Fundamentals of GNN

There are many ways to represent a graph mathematically. The most often used method is with the help of an adjacency matrix A. Graph with n nodes will have matrix A with dimension $(n * n)$. If in a graph, two nodes are connected then their corresponding matrix entry is filled with 1 or some weight if the edge is weighted. In other cases it would be zero. The nodes in the graph can have their own features f that define it. For example in case of a person as a node in social network graph, the features would be age, gender, country and so on. The node feature matrix for a graph will have dimensions $(n * f)$.

The main goal is to map each node to a lower dimensional space. This concept of mapping is known as Node Embedding. While doing this it is necessary to preserve the features and relationships of the original graph. Thus it is essential that the similarities between nodes in the original graph network are equal to the similarities between the embedded nodes in reduced dimension embedding space. Based on the embeddings of node, one can perform various graph learning tasks such as anomaly detection, clustering, classification and regression at node level and graph level, link prediction etc. To get the node embeddings, an ecoder function $ENC(u), ENC(v)$ is needed which converts the feature vectors of nodes x_u, x_v in input graph to new feature vectors z_u, z_v in the lesser dimensional embedding space while the similarity function could be euclidean distance. The encoder function needs to perform some specific sequential tasks in order to preserve, gather and transform the information from input graph to output node embeddings. This process can be observed in figure 4.9. The tasks to perform in this process are discussed next.

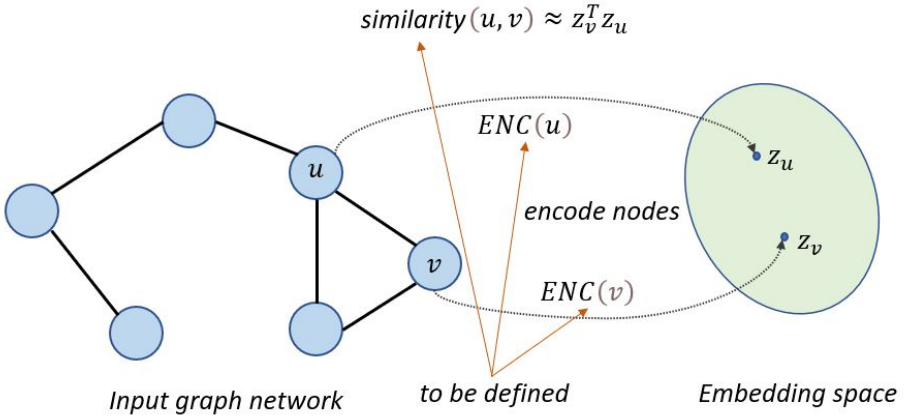

FIGURE 4.9: Node Embedding

Gather Locality Information

Locality information is knowing about the neighbourhood of a node in the graph. This information helps the network to learn about the range of nodes whose features affect the representation of a particular node in graph. For example, consider node i in the figure 4.10.

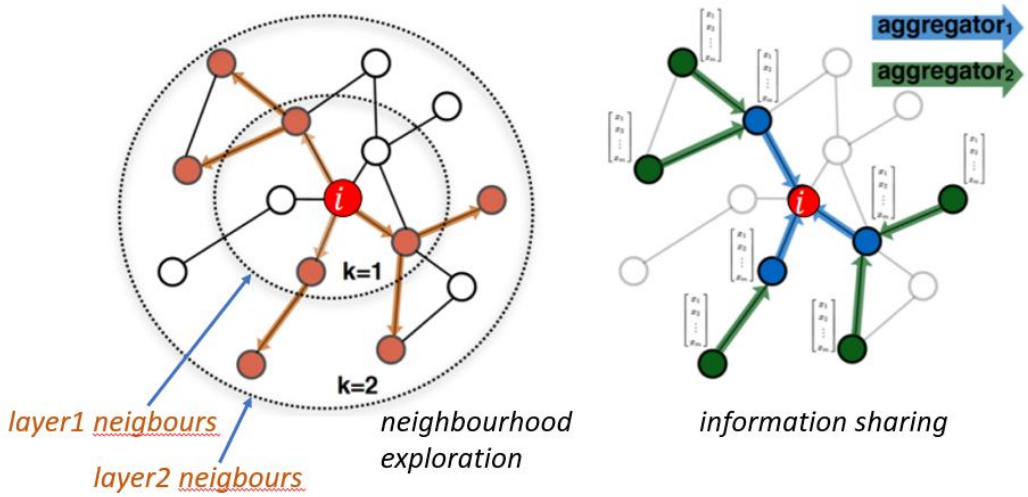

FIGURE 4.10: Gathering locality information[7]

Next step is to observe how node i is connected to its neighbors and neighbors of its neighbors. The resulting structure representing a part from the graph is known as computational graph. Each node has its own computational graph. It helps to identify where each of the node belongs in the graph. By building a computational graph it is possible to not only capture the structural information but also to borrow the feature information essential for a particular node.

Aggregate and Compute the Information

Once the computational graph is ready, the next step is to aggregate and transform the information. This is done with the help of neural networks represented by the boxes in the figure 4.11.

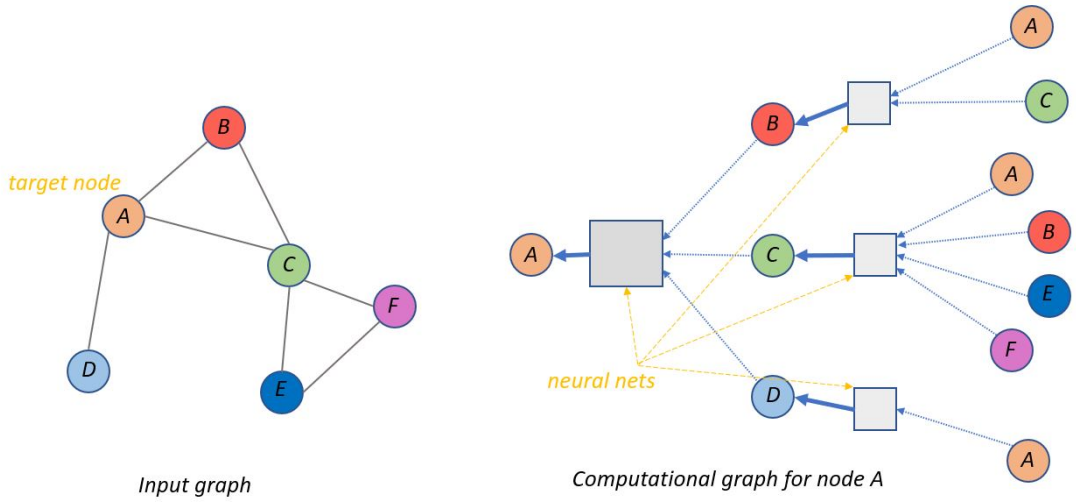

FIGURE 4.11: Input graph and a node computational graph[7]

The aggregation is done with the help of order-invariant aggregation functions for eg. sum, average or maximum. Once the methodology is defined, the next step is to go for the forward propagation in GNN. The forward propagation will decide the flow of information from input end to the output end of neural network. There are the three main steps to be performed for forward propagation on computational graph.

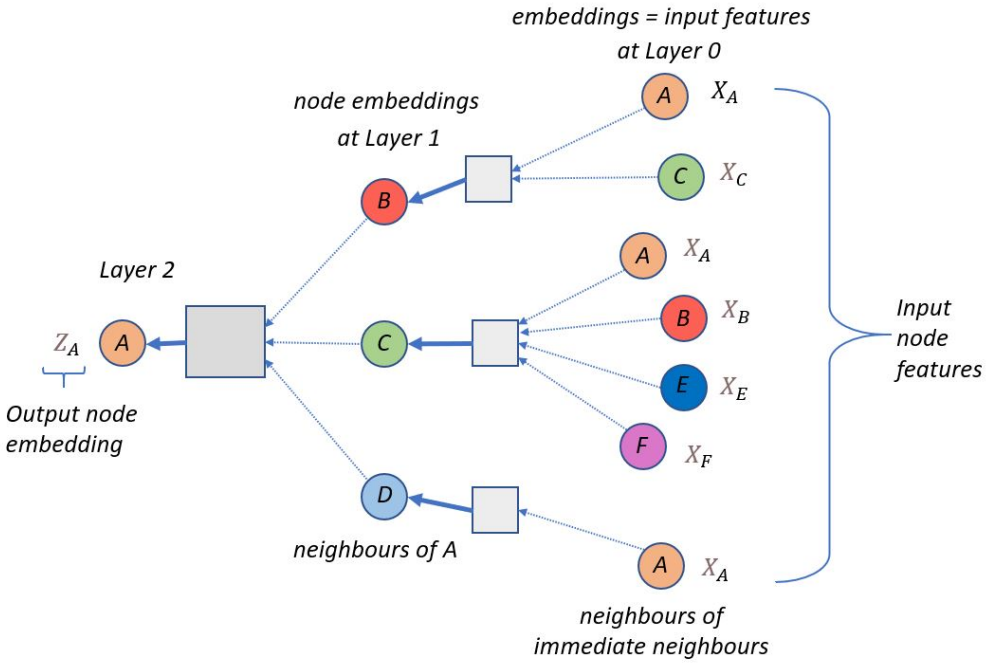

FIGURE 4.12: GNN deep model[7]

- Initialization with input node features (h_v^0): The input graph contains node features for every node. Consider X_A, X_B as input vector features for node A, B and so on. The nodes at layer 0 are initialized with their respective input feature vectors. Thus for any node v at layer 0 we can write,

$$h_v^0 = X_v \tag{4.8}$$

- Node embeddings in every layer: After the nodes in layer 0 are initialized with input feature vectors, the feature vectors are aggregated with the help of neural networks and passed on to next layer. For example, X_A and X_C are aggregated and passed on to node B in the figure shown above. For every node v in layer k the node embeddings can be written as

$$h_v^k = \sigma(W_k \sum \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}), \forall k \in \{1, ..., K\} \tag{4.9}$$

The above equation can be broken down into two parts. First part is about averaging of all the features of neighbours of node v as

$$(W_k \sum \frac{h_u^{k-1}}{|N(v)|})$$

W_k is weight matrix while h_u^{k-1} represents the node features from previous layer. The second part is the self-loop activation for node v

$$B_k h_v^{k-1}$$

B_k is the trainable bias matrix and h_v^{k-1} represents the embeddings of node from previous layer $(k-1)$. σ is the non-linear activation function applied, for eg. ReLU.

- Getting the final equation: The node embedding for node v obtained at the final layer K can be written as

$$z_v = h_v^K \tag{4.10}$$

After achieving our goal of obtaining the final node embeddings in reduced dimension space (latent representation) for every node, next process is to define a loss function on these embeddings and run stochastic gradient descent algorithm to train the parameters W and B.

4.2.3 General GNN Pipeline

The basic steps for building a GNN model include- (1) Defining the application oriented graph structure, (2) Specifying the graph type, (3) Designing the loss function, and (4) Building the model using computational modules.

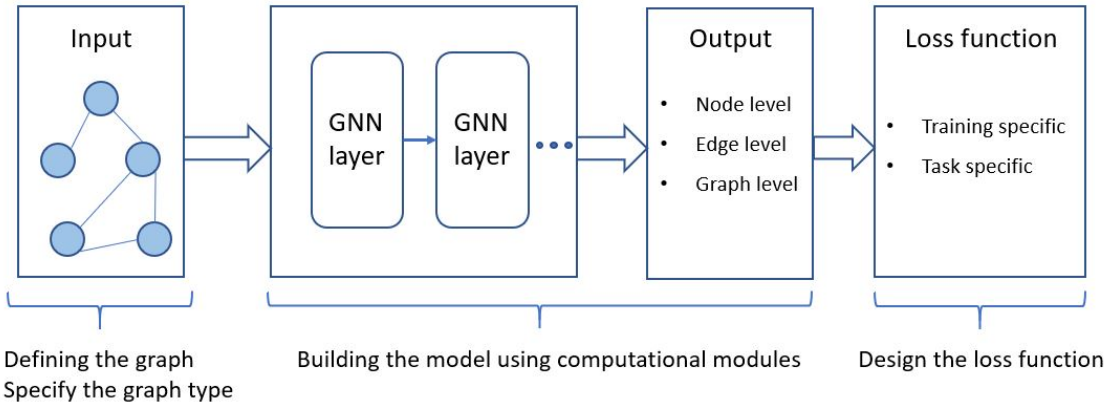

FIGURE 4.13: General GNN design pipeline[6]

Defining the graph structure

A graph structure is application oriented and can be classified in two scenarios- structural and non-structural. Structural scenarios involves explicit graph in the application, for example in case of molecules, knowledge graphs, physical systems and so on. In non-structural scenarios, the graphs are implicit and require the need to first build the task oriented graph for example, building a scene graph for an image or building a word graph for the text[6].

Specifying the graph type

After getting the application specific graph, the next step is to specify the graph type. Graphs can be categorized as- directed or undirected graph, homogeneous

or heterogeneous graph, static or dynamic graph. Edges in the directed graphs are directed from one node to other while an undirected graph can be considered as having bidirectional edges between the nodes. Homogeneous graphs have the same type of nodes and edges while heterogeneous graphs have different types of nodes and edges. Static graphs have input features and structure which do not vary over time while the dynamic graph have time varying topology and input features[6].

Designing the loss function

The loss function plays an important role in getting the desired output. It is therefore essential to know the task goals and the training settings to decide an appropriate loss function.

Graph learning tasks are of three types- node-level, edge-level and graph-level. Node-level tasks are node classification (categorize nodes into several classes), node regression (predict continuous value for each node), and node clustering (partitioning the node into groups). Edge-level tasks are usually edge classification (classify edge types), and link prediction (predicting the existence of edge between nodes). Graph-level tasks need the model to learn graph representations for classification and regression purpose[6].

Another factor for deciding the loss function is the training setting. Graph learning can be categorized based on different training settings like- supervised, semi-supervised and unsupervised. There is labeled data for training in supervised setting. The semi-supervised setting has small amount of labelled data for nodes or edges and the model needs to predict the unlabelled part. Most node and edge classification problems are semi-supervised. Node clustering is an example of unsupervised setting scenario where the data is unlabeled and the model has to find the patterns.

Knowing the task type and the training setting now enables us to design a specific loss function for example, in case of semi-supervised node level classification task, the cross entropy loss is a good choice and can be used for labelled nodes in the training set[6].

Using computational modules and building the model

The most often used computational modules for building the model are- propagation module, sampling module and pooling module. The propagation module propagates the information between nodes in such a way that the aggregated information captures both feature and topology. When the size of graph is large, sampling modules are used to conduct the propagation. The sampling module is generally combined with the propagation module. Pooling modules extract the information from nodes when representations of high level graphs or subgraphs are needed[6].

According to the pipeline discussed, in this thesis the first step would be to define the application oriented graph structure for the problem case scenario. Since the problem case will fall under the category of structural scenario of physical systems, the graph will be an explicit graph. The next step is to specify the graph type as undirected graph of homogeneous nature which implies single type of nodes and edges with bi-directional connection relation between the nodes. Since the thesis task is graph-level regression, it makes sense to experiment with loss functions like

Mean Absolute Error and Mean Squared Error. Finally the model will be built using propagation module that follows forward propagation method in GNN.

Chapter 5

Case Study Scenario

This chapter discusses the actual application of FEM and DNN techniques on a problem case scenario. A Cantilever Beam subjected to loading has been selected for the case study. Firstly, a general overview about the cantilever beam gives some familiarity with the problem case. The next step is to perform FEA of cantilever beam using Ansys software package to obtain the simulation results. Finally, deep learning methods are used to create a model architecture that would learn and predict the results confirming to the FEA.

5.1 Cantilever Beam

A cantilever beam is a rigid structural element which is supported at one end and free at other[54]. The support must be a fixed support to counter the forces and moments in all directions. The fixed end of cantilever beam is cast or anchored to a vertical support like a wall while the other end is free to carry loads. In the field of construction, a cantilever beam is often made of concrete or steel. It allows overhanging structures without bracing or additional supports.

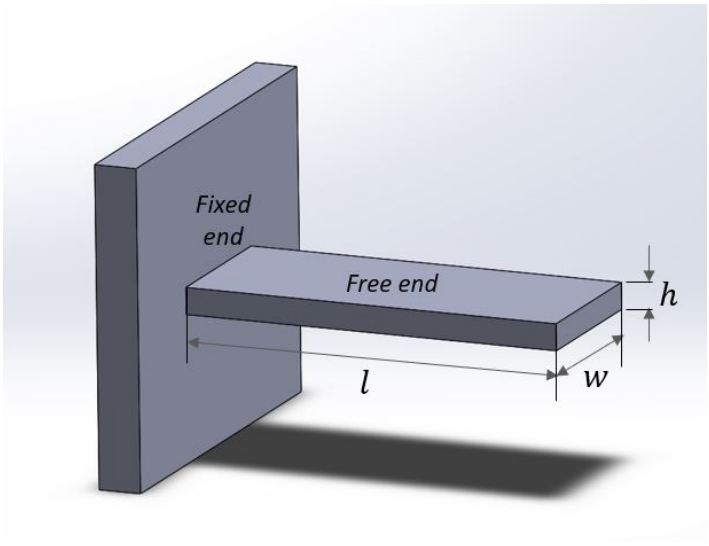

FIGURE 5.1: A cantilever beam

A cantilever beam is often subjected to point load, uniform load or varying load. In all the cases, it is subjected to bending because of the forces which creates tensile

stresses in the upper fibres and compressive stresses in the lower ones. Cantilever beams are usually constructed with trusses, slabs and racks for structural loading. Some of the most common areas of application can be found in construction of buildings in form of balconies or aesthetic roofs, construction cranes lifting heavy loads, vehicles parking shelter, wings of an aircraft and span of bridges.

5.2 Finite Element Analysis of Cantilever Beam

The FEA of the cantilever beam is done with the help of Ansys software package. The step wise procedure involves- generating 3D geometry of the beam, generating the finite element mesh, specifying the forces and boundary conditions and finally running the simulations to obtain the desired results.

5.2.1 Design of Cantilever Beam

The Ansys simulation package has a provision of designing parts or components with the help of modules like Space Claim and Design Modeller. The figure below shows the Design Modeller interface which has functionalities similar to any other design software. Ansys also has the provision of importing the designs created in other design softwares like Solidworks or Catia.

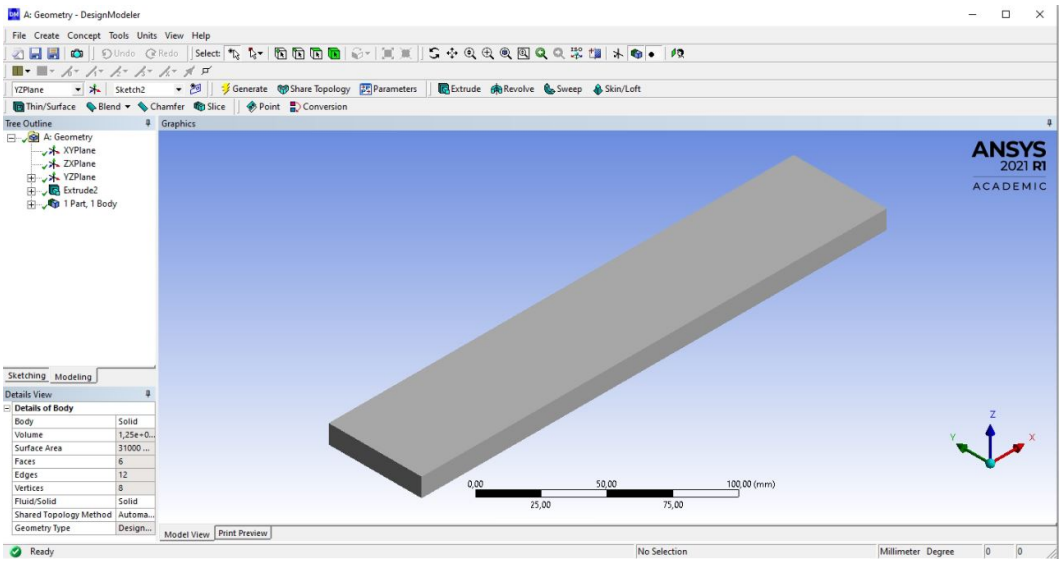

FIGURE 5.2: Ansys design modeller interface

In the design modeller interface we generate a rectangular sketch of dimensions $w = 50mm$ and $h = 10mm$ on the YZ plane. This section is then extruded by $l = 250mm$ in the X direction. This generates a cantilever beam in 3D which will be used in further analysis. From the engineering data book provided by Ansys we assign structural steel as the default material of the beam.

5.2.2 Generating Finite Element Mesh

After generating the design, the next step in FEA is to divide the continuous geometry into finite elements. This is where the created geometry is plugged in from design modeller into model definition. The descretization step that converts the geometry into collection of nodes and elements is achieved using the mesh settings in Ansys.

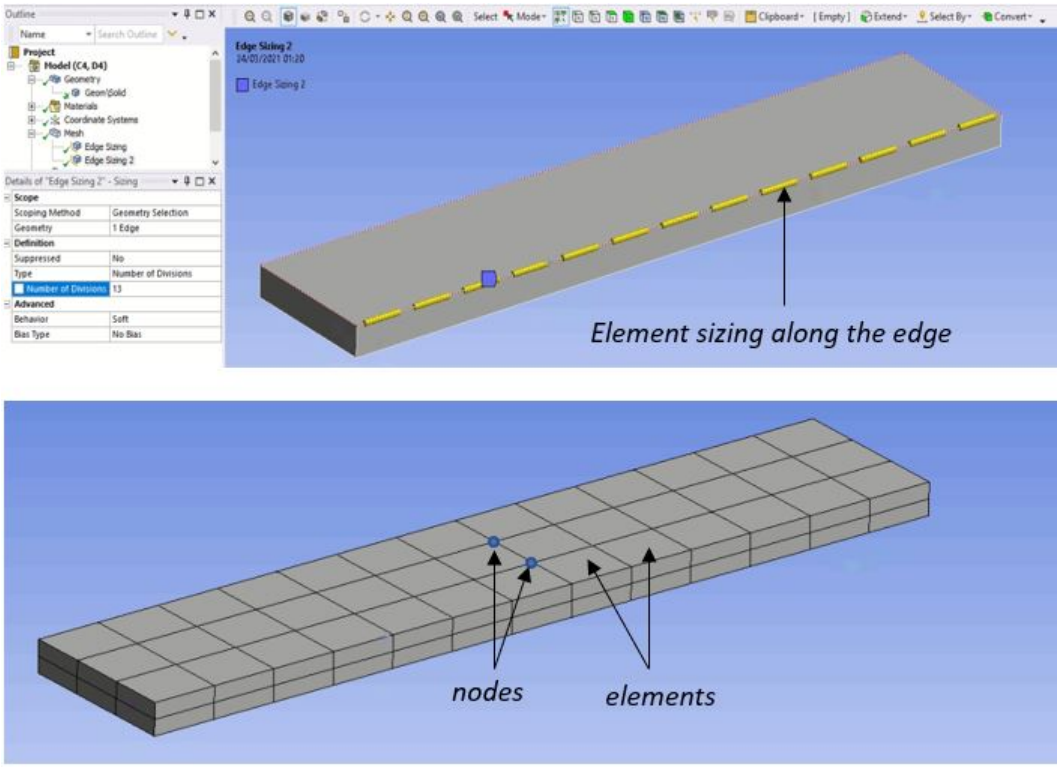

FIGURE 5.3: Meshing

The number of elements can be defined and controlled with the sizing feature along the edges. We specify 13, 3 and 2 as the default number of elements along l, w and h respectively. After the meshing is complete, the beam appears in form of a 3D grid where nodes are at the boundaries of the finite elements.

5.2.3 Specifying the Boundary Conditions

As the cantilever beam has one end fixed and other end free, it is necessary to specify the constraints in the software tool. This is done by using fixed support tool option on the surface to the left end. Applying fixed support ensures that there is no displacement in any direction of nodes at that end of the cantilever beam.

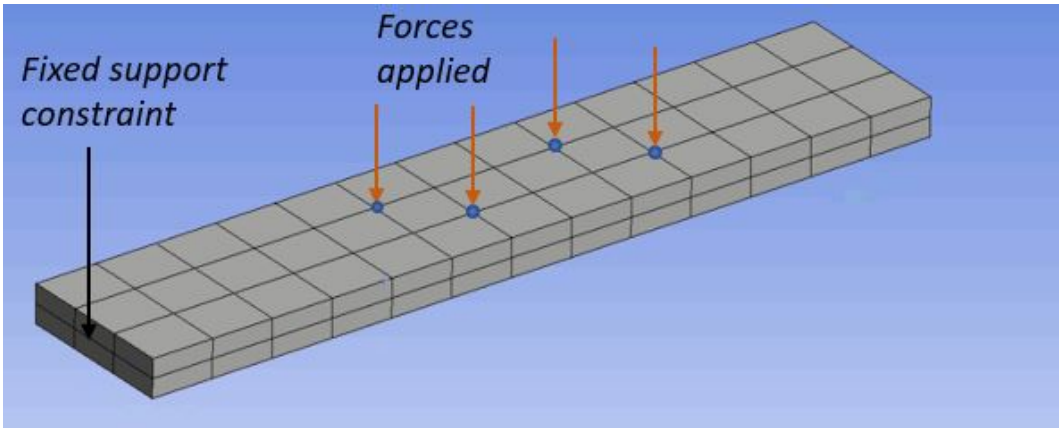

FIGURE 5.4: Boundary conditions

Once the fixed support is applied, the next step is to apply the forces at the free end of cantilever beam. In general the forces applied get distributed among the nodes connecting the finite elements. Therefore different nodal forces are applied at different nodal point locations. The default direction is kept along the Z axis in order to observe the bending of cantilever beam similar to any practical example. The magnitude of forces will be different at different nodal points indicating a varying load condition. Besides this, different set of loading conditions is specified at five different time steps to observe results at each respective time step analysis.

5.2.4 Obtaining the Results

After giving the sequence of different inputs for different time steps, the next step is to run the simulation and obtain the result for every case. Before that, it is essential to specify the types of output that we need to generate. A cantilever beam subjected to loading bends in downward direction. Hence, it would be interesting to see the deformation for different input force settings. Also the loading induces tensile and compressive stresses which might result into failure of cantilever beam if it exceeds certain limit. The limit is often defined by the yield strength point in the stress-strain curve of a material.

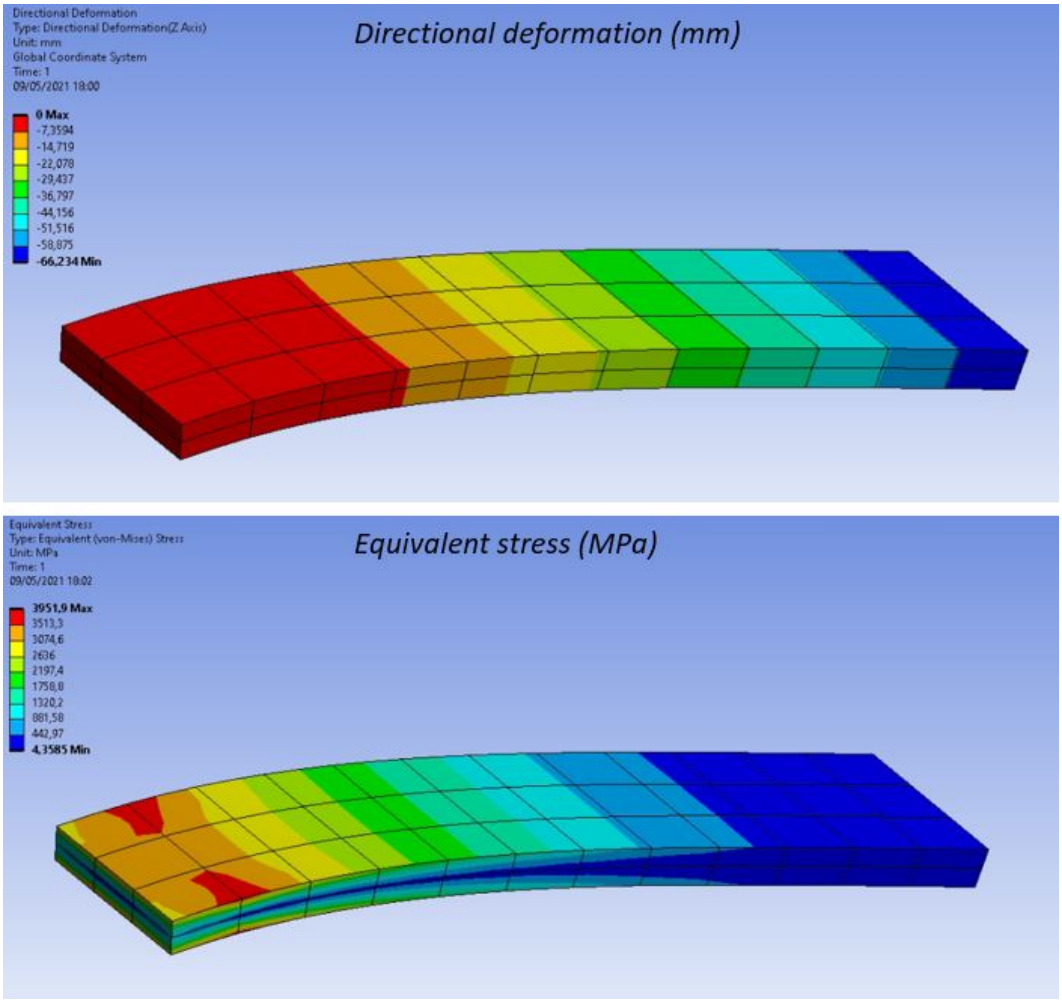

FIGURE 5.5: Output directional deformation and equivalent stress

The output is obtained in form of directional deformation along Z and equivalent (von-mises) stress. For this the nodal deformation and stress tools are selected from the solution toolbox to get the output values at every node. After this setting, the solver is run to obtain the solution for different input settings. A dataset of 2000 datapoints is generated by running multiple simulations. This dataset acts as a benchmark for training and testing the DNN model.

5.2.5 Dataset Generation

The inputs are the forces applied (N) along Z direction at every node location. The outputs are the deformation (mm) and stress values (MPa) at every node obtained as FEA result. The input is given in form of different settings of forces for five different time-steps and this sequence of forces goes in as single cell input entry of the dataset. Running the FEA solver for every time-step input gives the corresponding output sequence for deformation and stress. This completes one row of observation and we iterate the process to generate the dataset of approx. 2000 rows. The figure below gives an idea of the dataset structure.

F0	F1	F2
[-311.66, -175.66, -225.88, -28.69, -129.93]	[-392.83, -112.43, -206.27, -12.62, -173.2]	[-276.84, -185.28, -350.93, -4.88, -126.47]
[-26.08, -100.92, -195.18, -175.47, -268.06]	[-89.0, -163.64, -143.89, -161.68, -243.0]	[-32.01, -95.5, -115.06, -162.07, -324.87]
[-29.61, -73.51, -228.9, -127.01, -116.34]	[-1.82, -78.34, -151.77, -347.43, -349.97]	[-95.94, -72.72, -132.76, -49.93, -312.81]
[-122.22, -296.57, -93.78, -194.72, -288.79]	[-107.41, -147.39, -45.04, -243.75, -335.58]	[-6.89, -56.04, -71.23, -221.08, -238.86]
[-236.93, -197.1, -136.11, -182.5, -105.82]	[-123.09, -168.28, -120.73, -117.87, -228.26]	[-282.48, -171.02, -127.77, -178.43, -2

inputs

D0	D1	D2
[-0.906, -0.546, -0.763, -0.094, -0.55]	[-1.138, -0.693, -0.961, -0.12, -0.701]	[-1.129, -0.693, -0.954, -0.12, -0.701]
[-0.184, -0.46, -0.556, -0.638, -0.951]	[-0.238, -0.587, -0.705, -0.812, -1.218]	[-0.241, -0.587, -0.704, -0.812, -1.223]
[-0.183, -0.172, -0.704, -0.71, -0.815]	[-0.234, -0.218, -0.895, -0.894, -1.031]	[-0.235, -0.218, -0.894, -0.891, -1.029]
[-0.736, -0.62, -0.176, -0.744, -0.725]	[-0.94, -0.789, -0.222, -0.944, -0.929]	[-0.944, -0.792, -0.221, -0.943, -0.9
[-0.675, -0.565, -0.551, -0.553, -0.686]	[-0.857, -0.717, -0.698, -0.707, -0.881]	[-0.857, -0.717, -0.697, -0.709,

S0	S1	S2
[5315.887, 3255.035, 4513.911, 564.652, 3293.723]	[4672.592, 2860.283, 3961.214, 496.971, 2897.793]	[4676.98, 2859.661, 3956.161
[1124.864, 2767.438, 3312.232, 3818.488, 5738.246]	[994.292, 2435.46, 2913.313, 3360.259, 5051.964]	[994.638, 2434.878, 2913.512
[1102.121, 1024.92, 4203.315, 4156.199, 4827.545]	[968.305, 902.884, 3695.858, 3657.366, 4240.236]	[968.317, 904.218, 3696.28
[4417.662, 3693.33, 1037.213, 4431.89, 4361.049]	[3887.596, 3252.321, 912.621, 3895.809, 3846.517]	[3885.478, 3254.611, 9
[4006.389, 3370.879, 3284.227, 3320.364, 4153.713]	[3520.676, 2965.518, 2886.096, 2924.41, 3662.499]	[3522.022, 2965.3,

outputs

FIGURE 5.6: Dataset structure

5.3 DNN Adaptation to Cantilever Beam Scenario

This section explains the application of deep learning methods, specifically deep neural networks and the combination of various architectures in order to predict the outcome of the cantilever beam case scenario comparable to the FEM method. The expressive power of different types of graph networks will be used to get a latent representation which we also call as node embeddings and later combine this with stacked linear layers to bring the output in required form.

5.3.1 Generating the Graph

The very first step in GNN pipeline is to generate the application oriented graph. In our case of cantilever beam, the structural analogy of FEM is used, where the beam is divided into discrete elements connected by nodes. Since the type of nodes and edges is constant, the graph will be homogeneous graph. To build a graph the four essential requirements are- node labels, source node, destination node and type of edge relation (directed or undirected). For the node labelling the integer number system is used, where every node is represented by a unique number just like in FEM method.

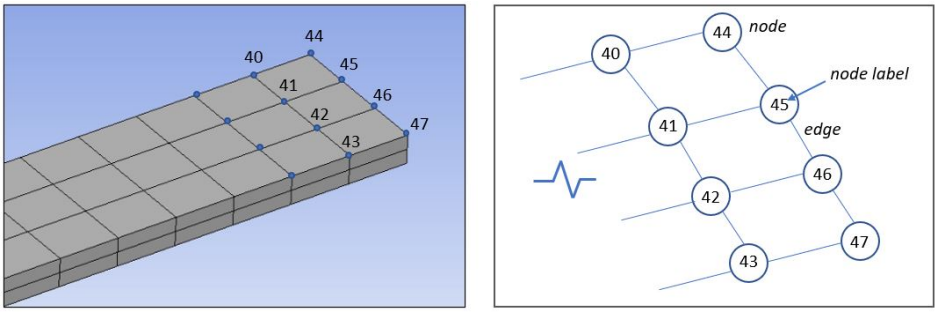

FIGURE 5.7: Generating graph using integer label for nodes

The edges are bi-directional in nature allowing mutual exchange of information. After mapping all the nodes from FEM to the GNN source and destination nodes, the graph structure is complete.

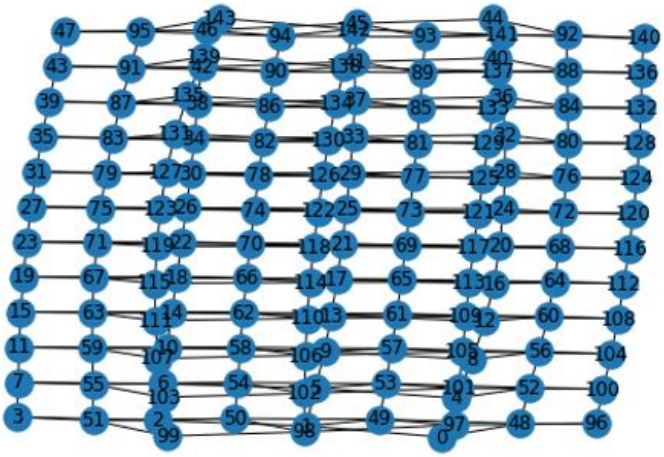

FIGURE 5.8: Graph network

The graph structure generated as shown in figure is equivalent to the grid connectivity observed in the 3D meshing of FEM. The next step is to build the neural net using different computational modules.

5.3.2 Building the Neural Network Model

The application specific graph structure generated is the input graph for further process. By using the generated graph structure, the locality information of each node is gained by building the computational graph as discussed in the 'fundamentals of GNN' section. The input graph is initialized with feature vector X as the sequence of force values applied at every node. We now move towards forward propagation in GNN where our aim is to obtain the node embeddings or the latent representation of nodes by aggregating and transforming the input data. As discussed before, the general equation for node embedding h for a node v in layer l can be written as-

$$h_v^k = \sigma(W_k \sum \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}), \forall k \in \{1, ..., K\} \tag{5.1}$$

Depending on the method of aggregating the information we have numerous GNN models with 'Graph Convolution Network' and 'Gated Graph Convolution' being some of the well known ones. According to [55] aggregation can be performed efficiently using sparse matrix operations transforming the above equation into-

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} A \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{5.2}$$

where, $\tilde{A} = A + I_N$ is the adjacency matrix with added self loops, $\tilde{D}$ is the diagonal matrix indicating the number of edges connected to a node and $W^{(l)}$ is trainable

matrix for a layer. σ represents activation function like ReLU. Gated Graph Convolution model on the other hand uses gated recurrent units favourable to graph-structured sequence based models[56]. The equations for propagation model according to [57] are-

$$m_i^{(l+1)} = \sum_{j \in N(i)}^{n} \Theta h_j^{(l)} \tag{5.3}$$

$$h_i^{(l+1)} = GRU(m_i^{(l+1)}, h_i^{(l)}) \tag{5.4}$$

Propagating the information through multiple layers helps in better representation learning as with every layer the neighbourhood area explored increases and so does the information gained by every node to determine where it belongs in the graph. We thus experiment with multiple layers and different architectures to observe the variation in end results.

The forward propagation in GNN models will yield the node embeddings at the final layer. To further improve the learning and also to transform the embeddings into required shape of output. The representations are passed further to multiple neural network layers (MLP). Experiments will be performed with non-linear activation functions like ReLU and Leaky-ReLu in both GNN and ANN stages of neural nets. The final output obtained is the deformation and stress values for each node for every time step force setting. The predicted output is compared with the actual output with the help of loss function to calculate the error for backpropagation. Different types of loss functions like MAE, MSE, L1loss will be experimented with to get the best results. The error is backpropagated and with the help of different algorithms used for different optimizers like RMSprop, Adam the weights are updated. The entire process is run for multiple batches of data in the dataset and multiple epochs are run to minimize the loss. To summarize, the flow of DNN approach is as shown in the figure 5.9.

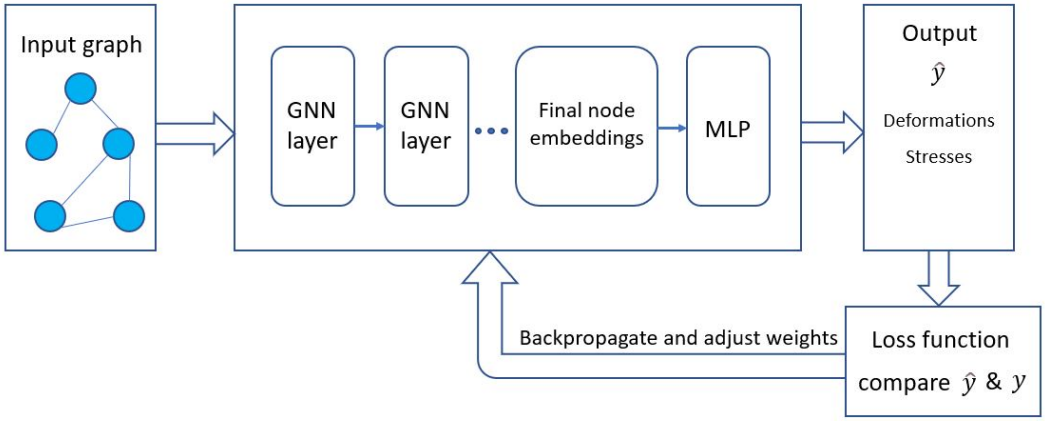

FIGURE 5.9: DNN model flowchart

Chapter 6

Experiments and Results

The aim of the work depicted in this chapter is to observe the effectiveness and adaptability of different DNN architectures to the variations in the FEA of cantilever beam. The FEA variations include- Tuning with the mesh sizing for more accurate results and generalizing the deformation result in form of twisting in addition to bending of cantilever beam due to imbalance of forces. Different DNN architectures will be trained on the datasets generated individually for the FEA variations to observe the effect of various factors like message passing strategies in graph network, number of layers and activation functions on the final outcome in form of predicted deformation and stress values and their closeness to the actual values from FEA software.

6.1 Beam with Coarse Mesh

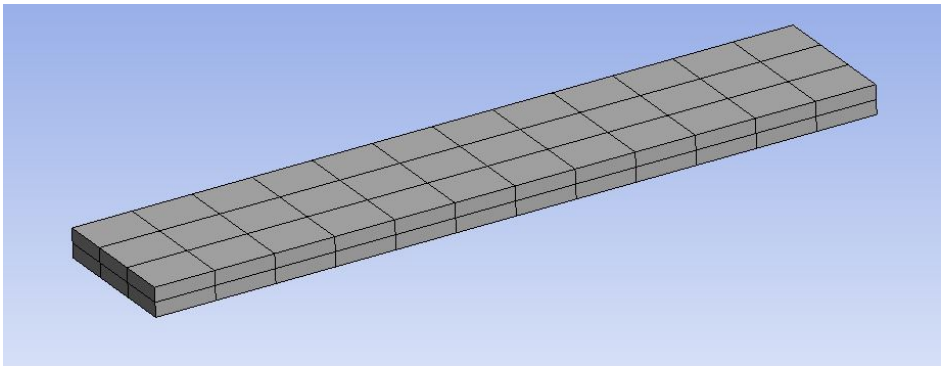

FIGURE 6.1: Cantilever beam with coarse meshing

The starting step in general FEA of a system is to obtain and observe the results for a coarse mesh setting. The results from coarse mesh funnels an immediate strategy for possible improvements in process of design or manufacturing. The computation time for coarse mesh setting is low, compromising with the accuracy of results generated. It is useful for projects that require quick deployment and small potential changes which are well within the threshold of safety and functionality.

The coarse mesh setting for the cantilever beam is as shown in the figure above. There are total 72 elements and 156 nodes. The beam is subjected to different point loads in vertically downward directions at the nodal points causing it to bend downward. The FEA solver is run to get the deformation and stress outputs at the nodes

for a sequence of force settings for five different time steps. This gives one row of observation, and the process is iterated to generate the dataset of 2000 rows.

6.1.1 Analysis with GCN Architecture for Coarse Mesh

The GCN architecture from [55] is based on an effective variant of convolutional neural networks operating directly on graphs. The model could be suitable for the considered case study scenario because it is highly efficient in learning the hidden layer representations that encode both- the structure of graph as well as node features. It follows the following propagation rule for a multi-layer graph convolutional network which was discussed in subsection 4.2.2 of chapter 4.

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}A\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{6.1}$$

We use the expressive power of graph networks by using 3 GCN layers to obtain the learned node embeddings and later pass these embeddings to a 2 layer MLP that transform this information to required form of output dimension. The architecture is depicted in the figure 6.2.

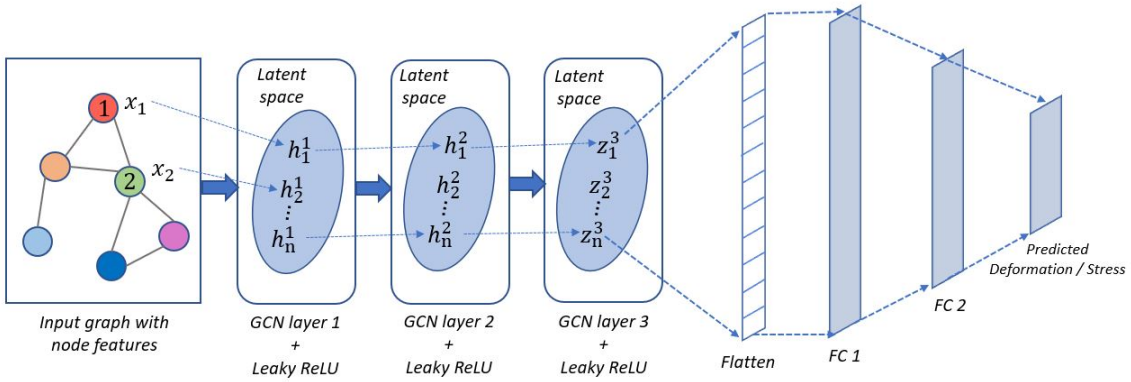

FIGURE 6.2: Model with GCN architecture

The above model is trained for 300 epochs with the best tuned hyperparameters found using the optuna package. The trained model is then made to predict the output for the test set data containing 300 rows of observations.

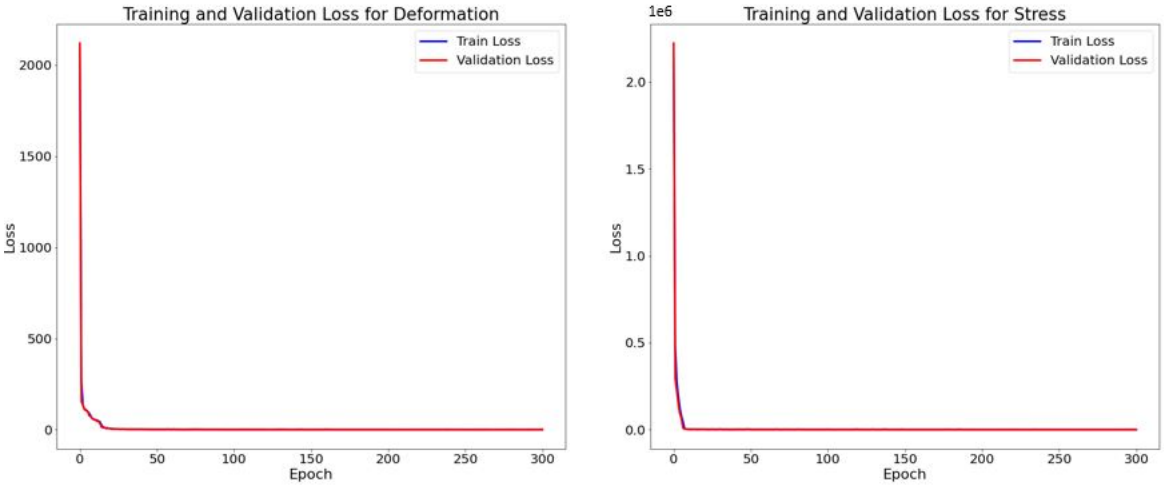

FIGURE 6.3: Training and validation curve for model with GCN architecture

While the individual results for the observation rows (data points) in the test set would vary, for displaying the results that would best represent the performance on test data, the most probable output is selected by determining the data point that gives loss value closest to the mean loss value for the test dataset. The loss function selected for this purpose is the L1 loss which gives the mean absolute error. Figure 6.4 gives an overview of this process.

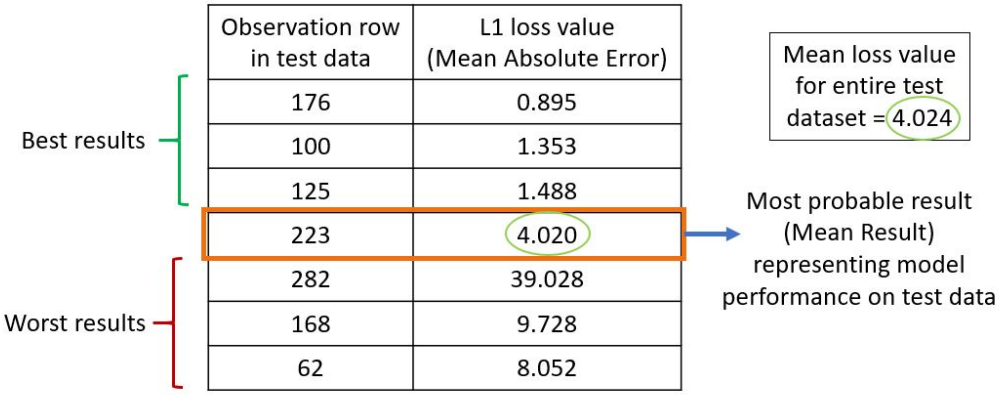

FIGURE 6.4: Selection of most likely result given by model on test data

The deformation result from DNN is visualized using 3D plots showing the difference between actual deformed beam in red shade and the predicted deformed beam in blue shade as shown in figure 6.5.

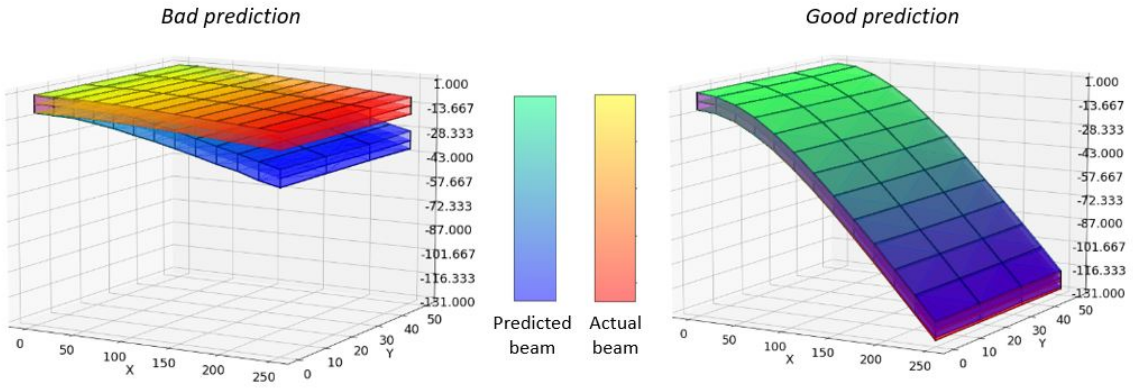

FIGURE 6.5: An example of DNN result visualization for deformation

The red beam comprises of the actual output deformation values obtained from FEA. The goal of the model is to predict these deformations with as minimum error as possible. When this happens, the two beams should appear to be overlapping. The results for the deployed DNN model with GCN architecture are shown next.

3D Plots for Deformation

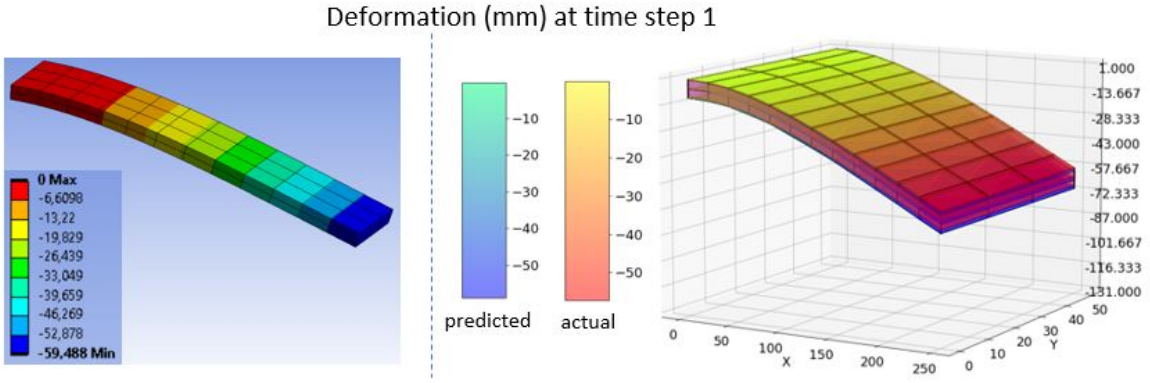

FIGURE 6.6: FEM vs. GCN result for deformation at time step 1

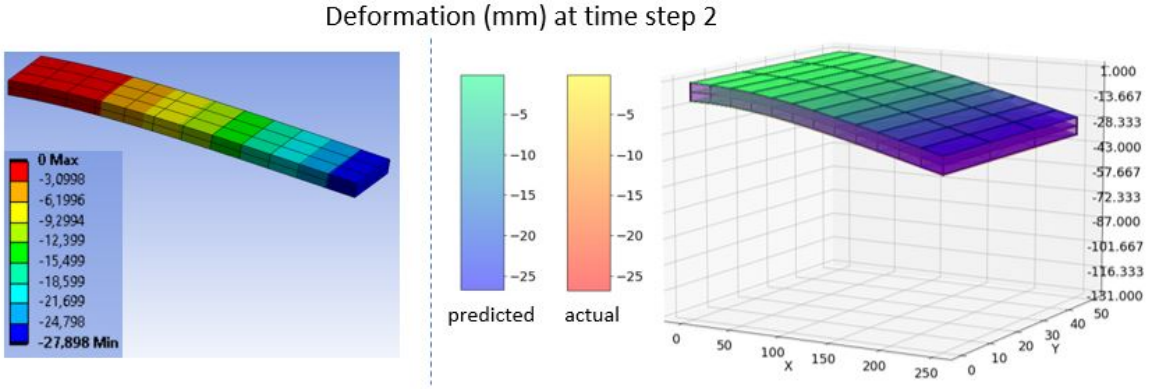

FIGURE 6.7: FEM vs. GCN result for deformation at time step 2

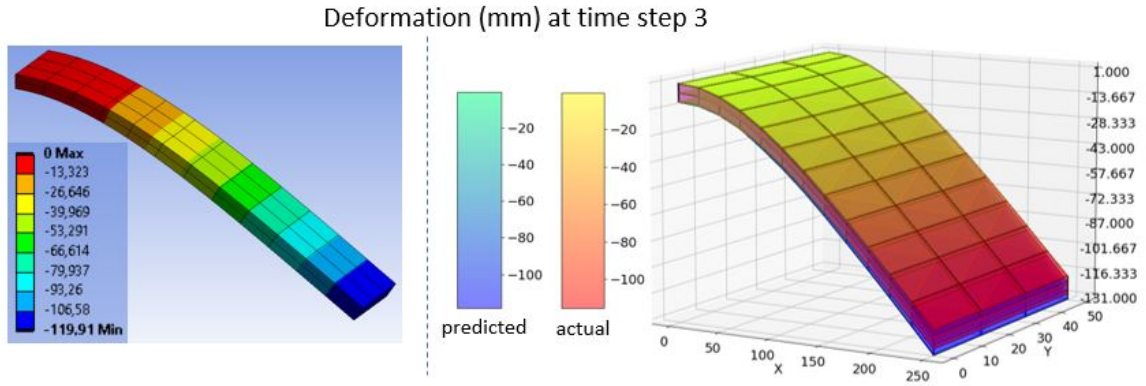

FIGURE 6.8: FEM vs. GCN result for deformation at time step 3

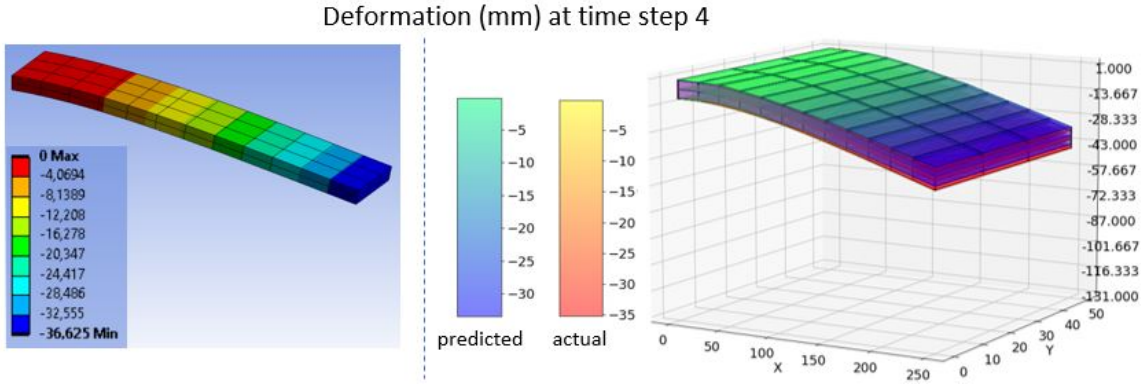

FIGURE 6.9: FEM vs. GCN result for deformation at time step 4

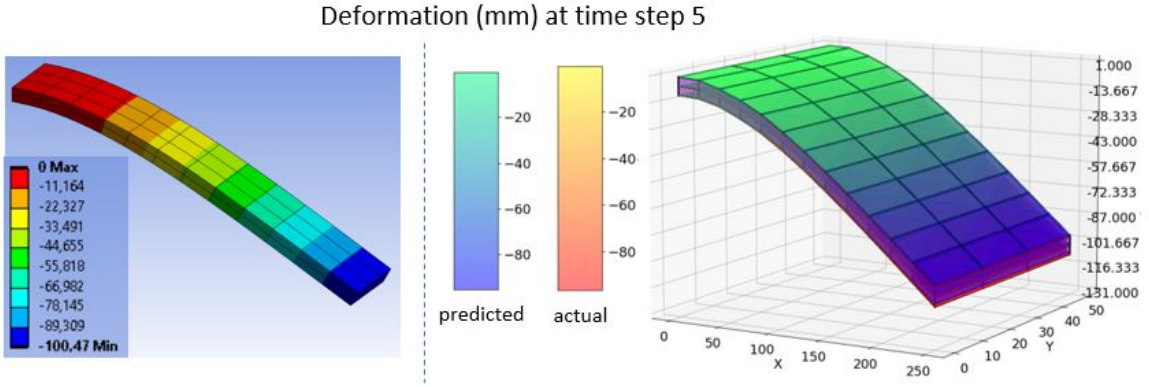

FIGURE 6.10: FEM vs. GCN result for deformation at time step 5

Figure 6.6 to 6.10 show the actual deformation results from FEA software on the left and the model predicted results on the right for five sequential time steps (each having different force setting as inputs). The overlapping of red beam (actual values from FEA result) and blue beam (model predicted deformation values) indicate highly accurate results.

Plots for Nodal Deformation Values

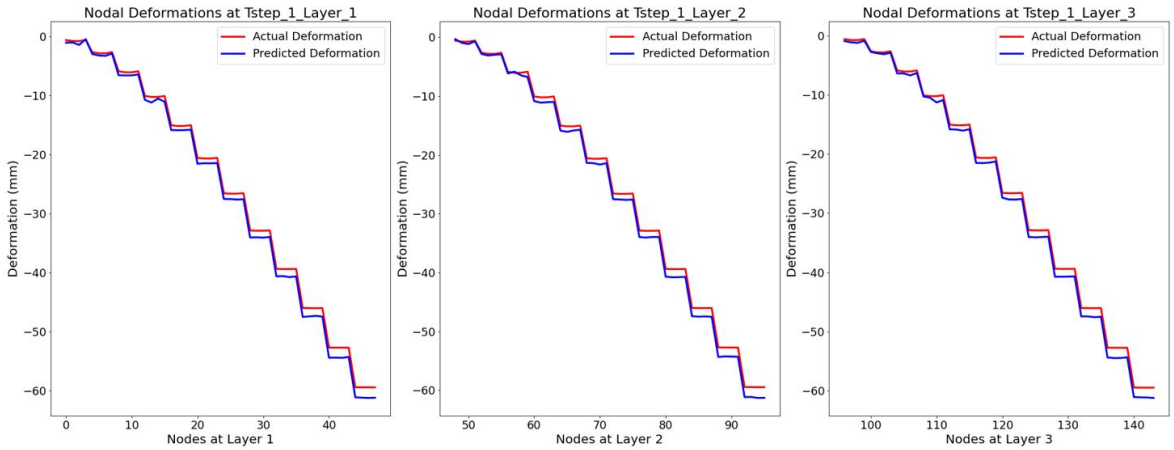

FIGURE 6.11: Nodal deformations for time step 1

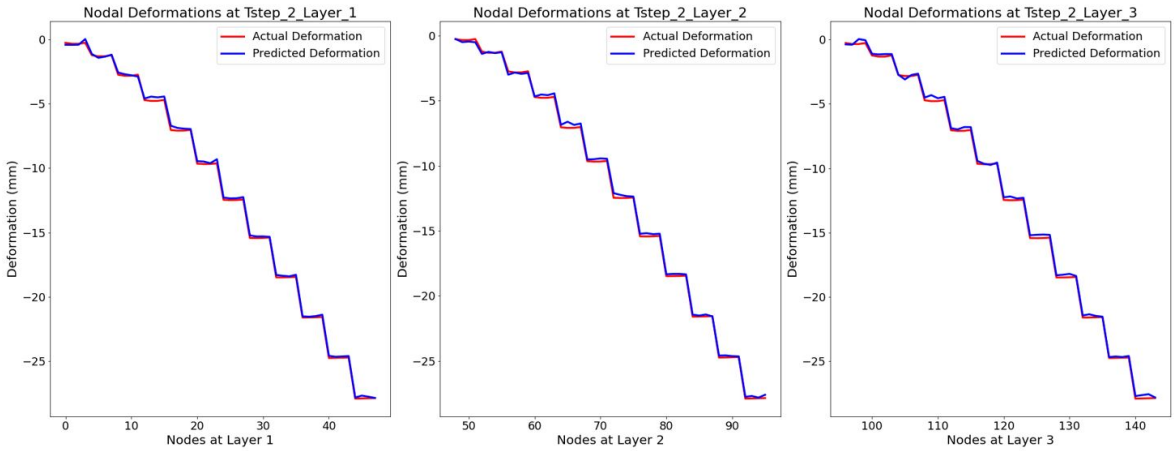

FIGURE 6.12: Nodal deformations for time step 2

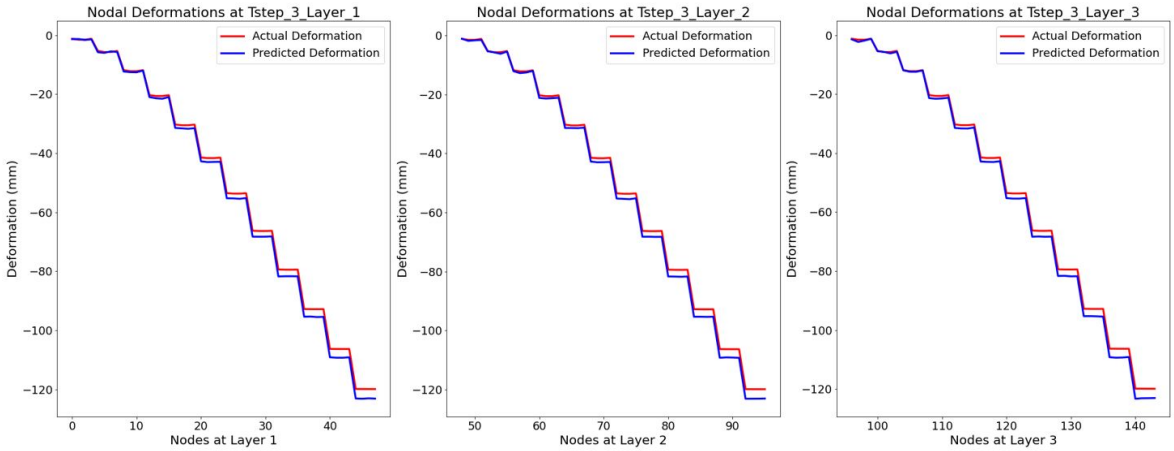

FIGURE 6.13: Nodal deformations for time step 3

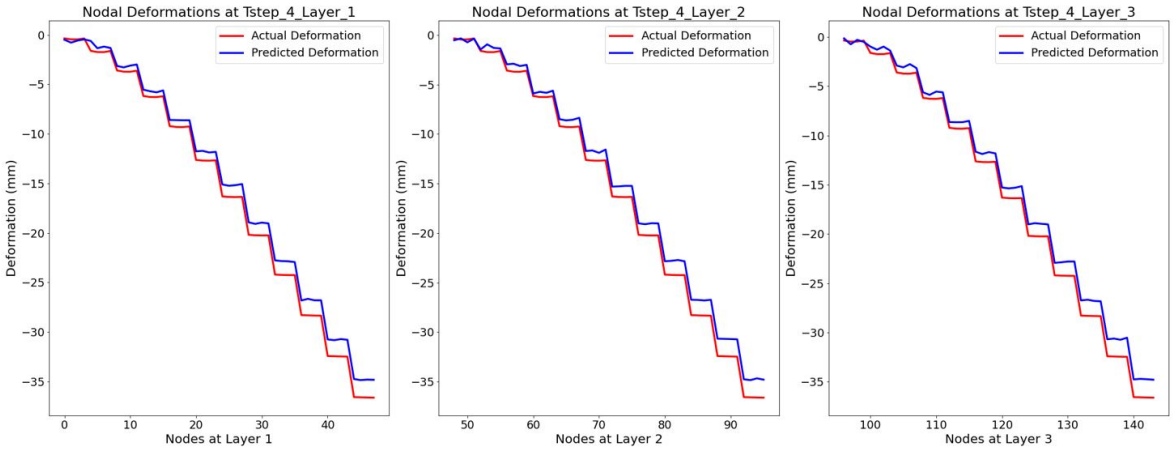

FIGURE 6.14: Nodal deformations for time step 4

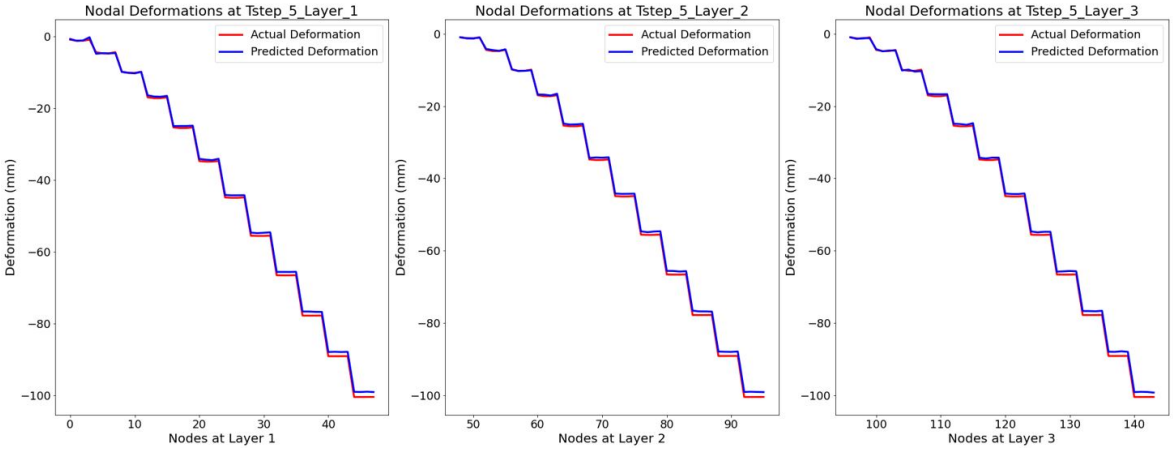

FIGURE 6.15: Nodal deformations for time step 5

Figure 6.11 to 6.15 show the Z deformation of nodes on cantilever beam for five sequential time steps. The nodes can be divided into thee layers- the top layer nodes, the middle layer nodes and the bottom layer nodes. While the 3D deformation plots give good visualization, the nodal deformation plots give actual in detail values for each nodal point.

Deformation Error Visualization

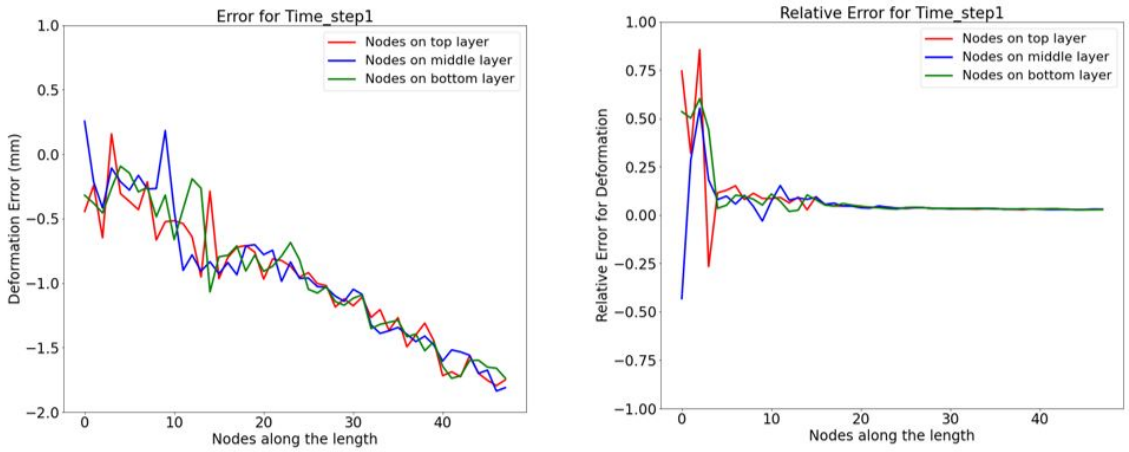

FIGURE 6.16: Error and relative error for deformation at time step 1

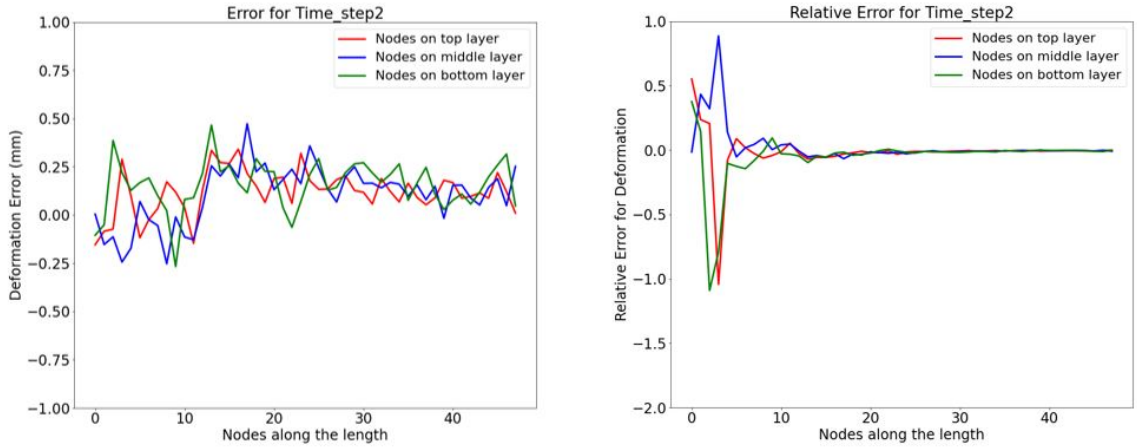

FIGURE 6.17: Error and relative error for deformation at time step 2

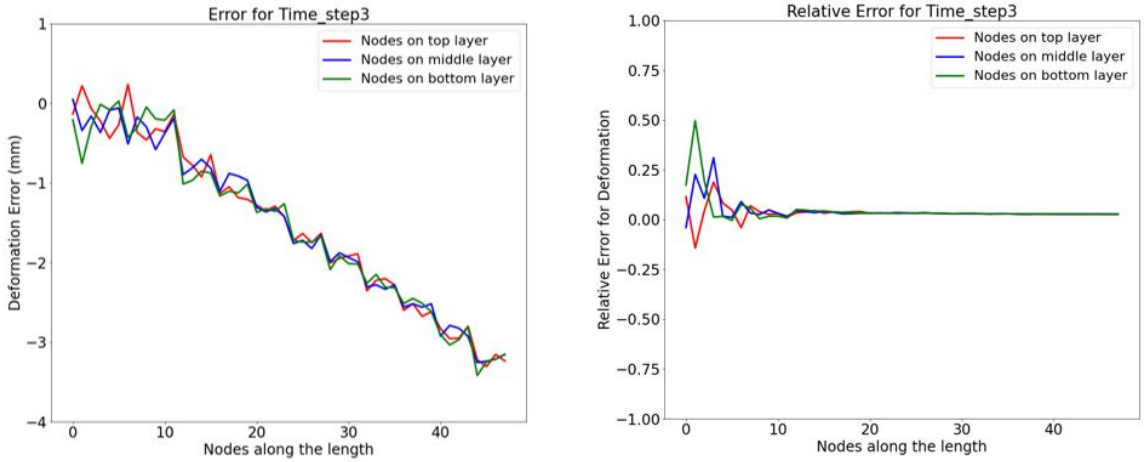

FIGURE 6.18: Error and relative error for deformation at time step 3

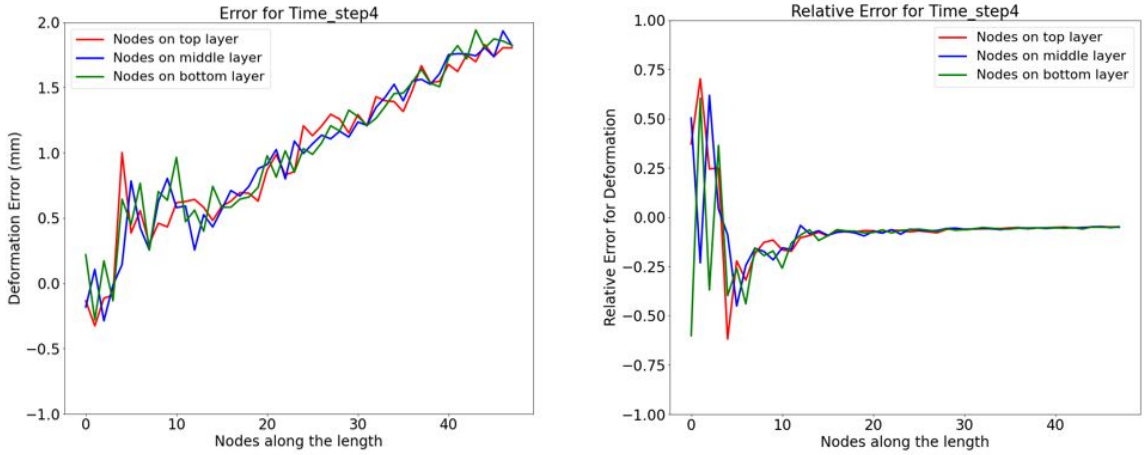

FIGURE 6.19: Error and relative error for deformation at time step 4

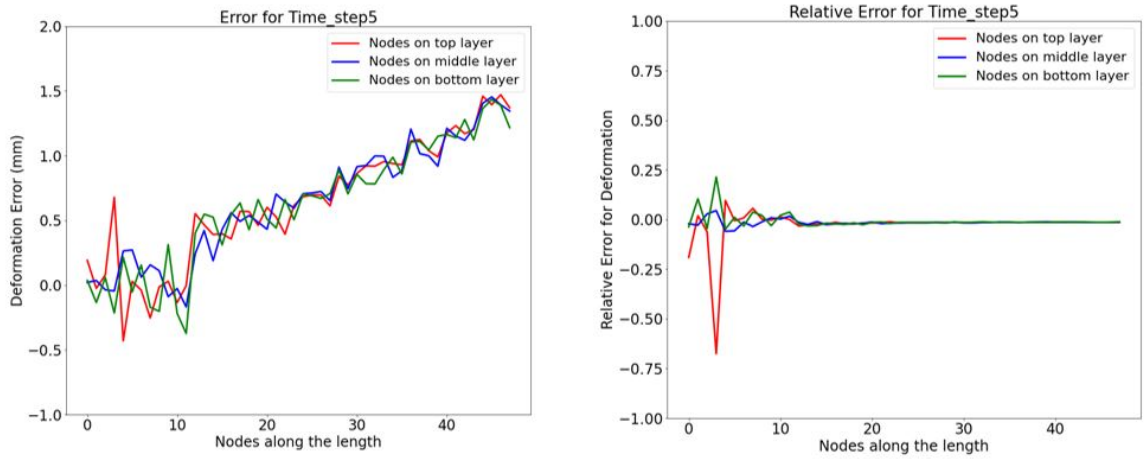

FIGURE 6.20: Error and relative error for deformation at time step 5

The plots on the left in figure 6.16 to figure 6.20 show difference between the actual deformation values and predicted deformation values for the nodes, while the plots on right side show the relative error specifying the error significance with respect to actual value. The early spikes in relative error plots are because of deformation values close to zero making even a small error look significant compared to the actual value. The maximum deformation error close to 2mm was seen in time step 4 error plot.

3D Plots for Stress

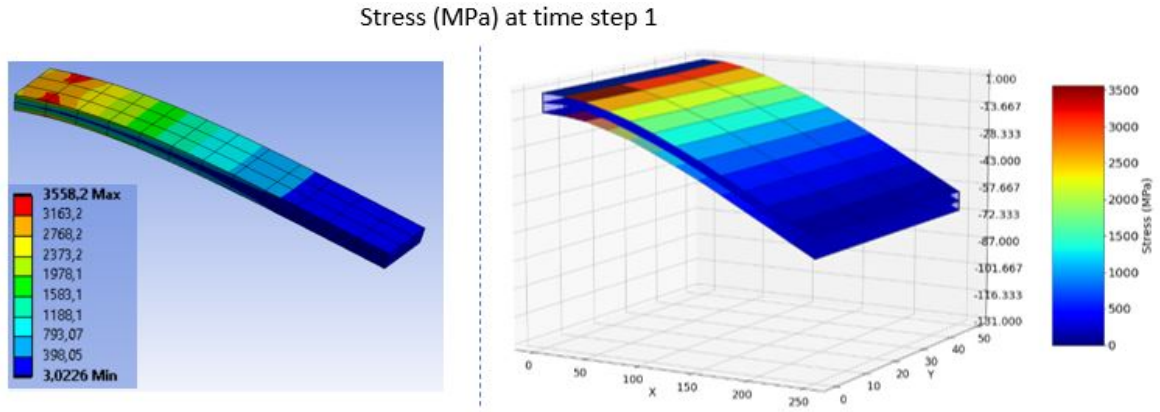

FIGURE 6.21: FEM vs. GCN result for stress at time step 1

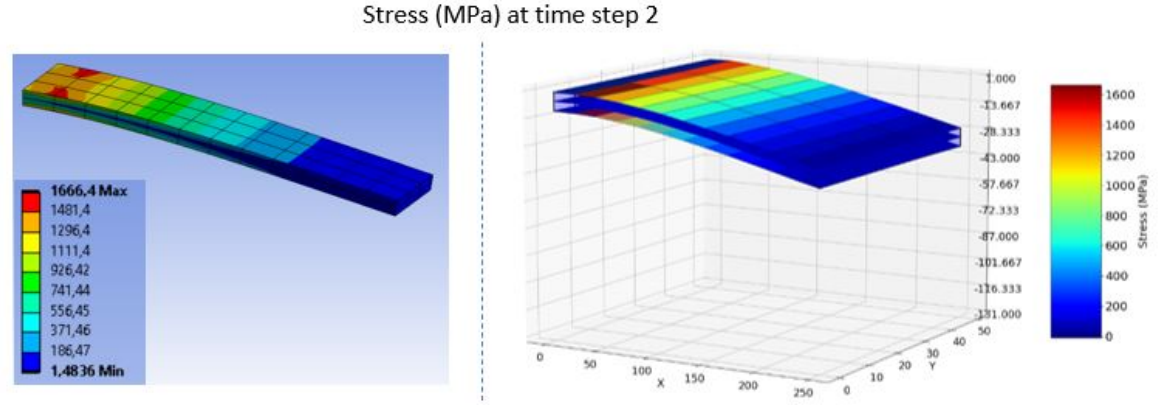

FIGURE 6.22: FEM vs. GCN result for stress at time step 2

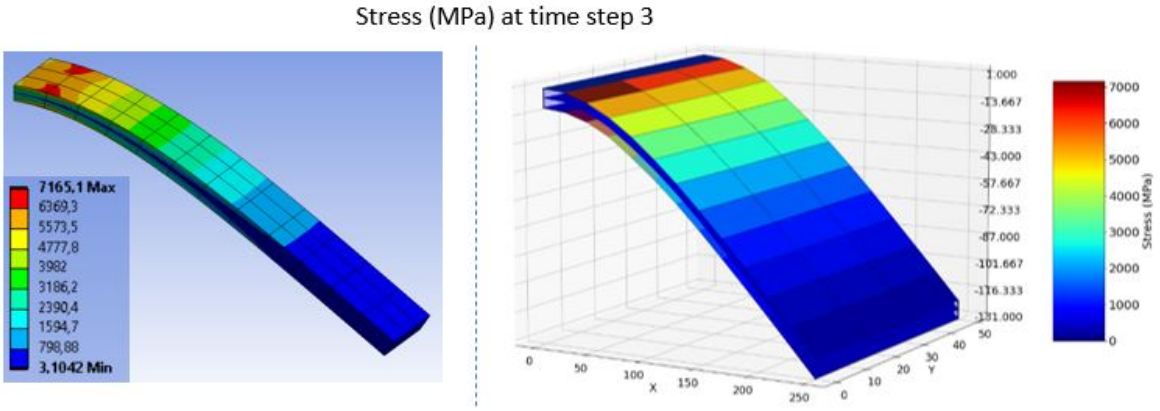

FIGURE 6.23: FEM vs. GCN result for stress at time step 3

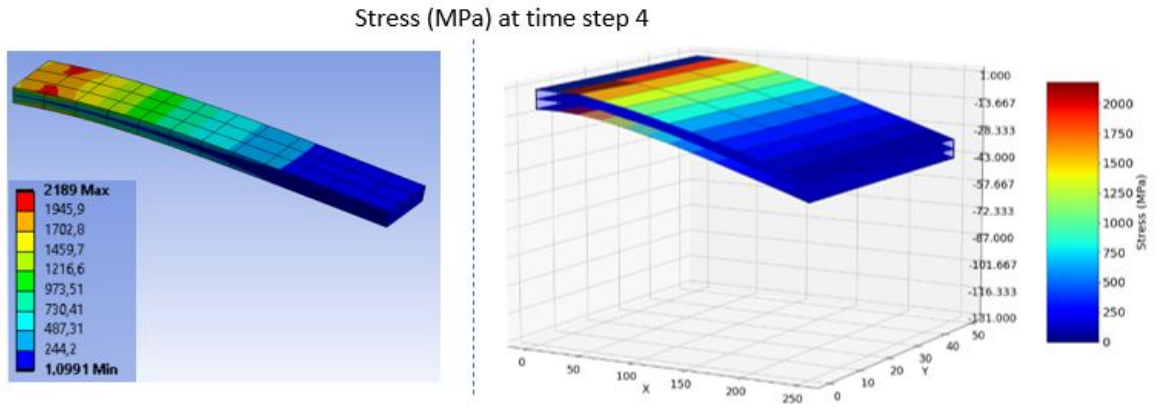

FIGURE 6.24: FEM vs. GCN result for stress at time step 4

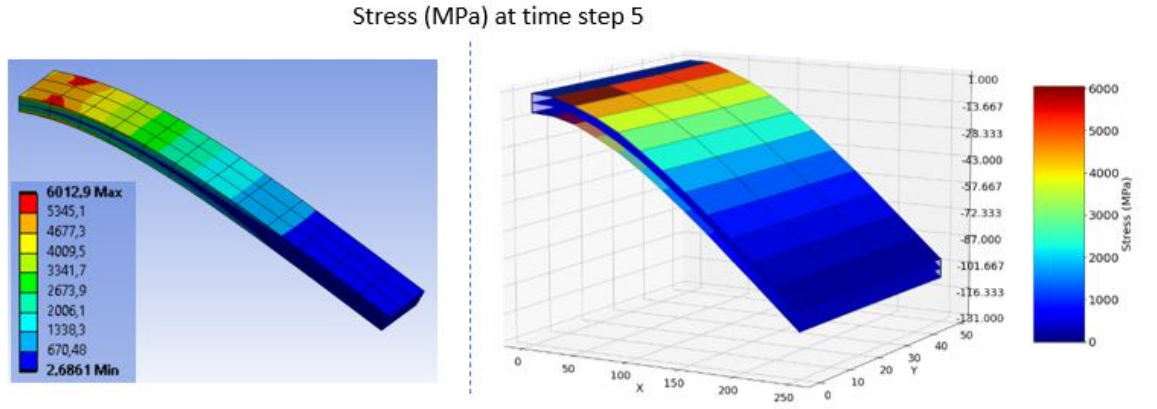

FIGURE 6.25: FEM vs. GCN result for stress at time step 5

Figure 6.21 to 6.25 show the comparison of stress visualization from FEA result (left) and model predicted result (right). The stress intensity is specified on the colorbars in both cases. The FEA software having more developed and sophisticated visualization method shows slight different color distribution on the beam as compared to the matplotlib 3D plot of beam on the right. Although the visuals differ slightly, the values when compared on the colorbars are identical.

Plots for Nodal Stress Values

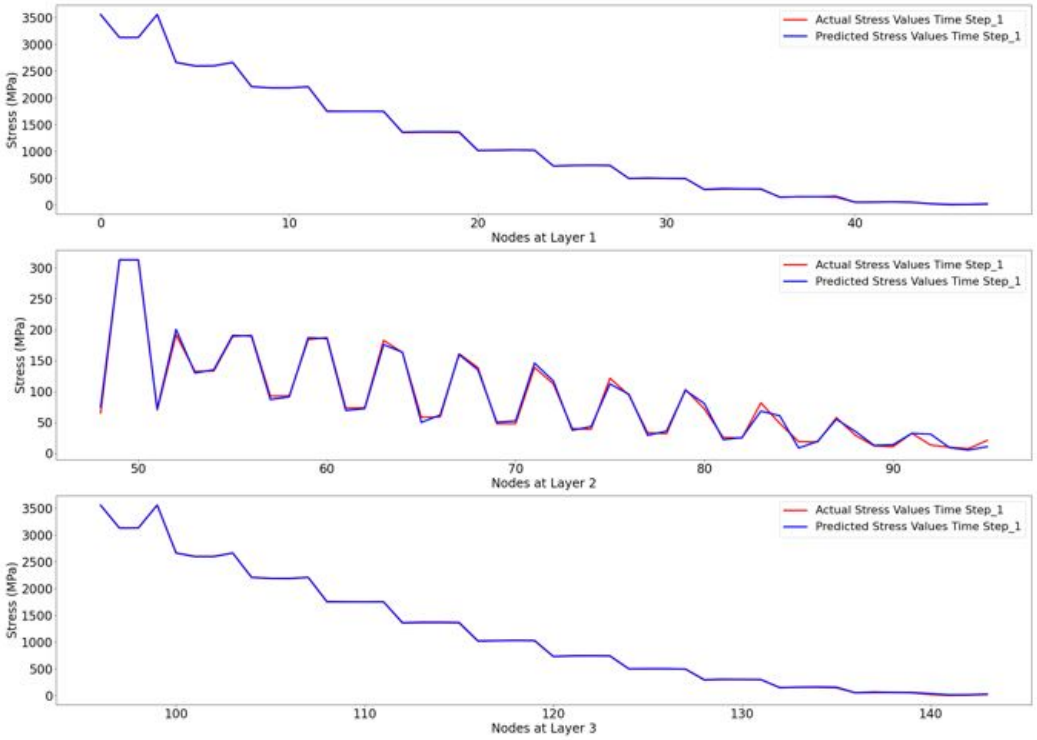

FIGURE 6.26: Nodal stresses for time step 1

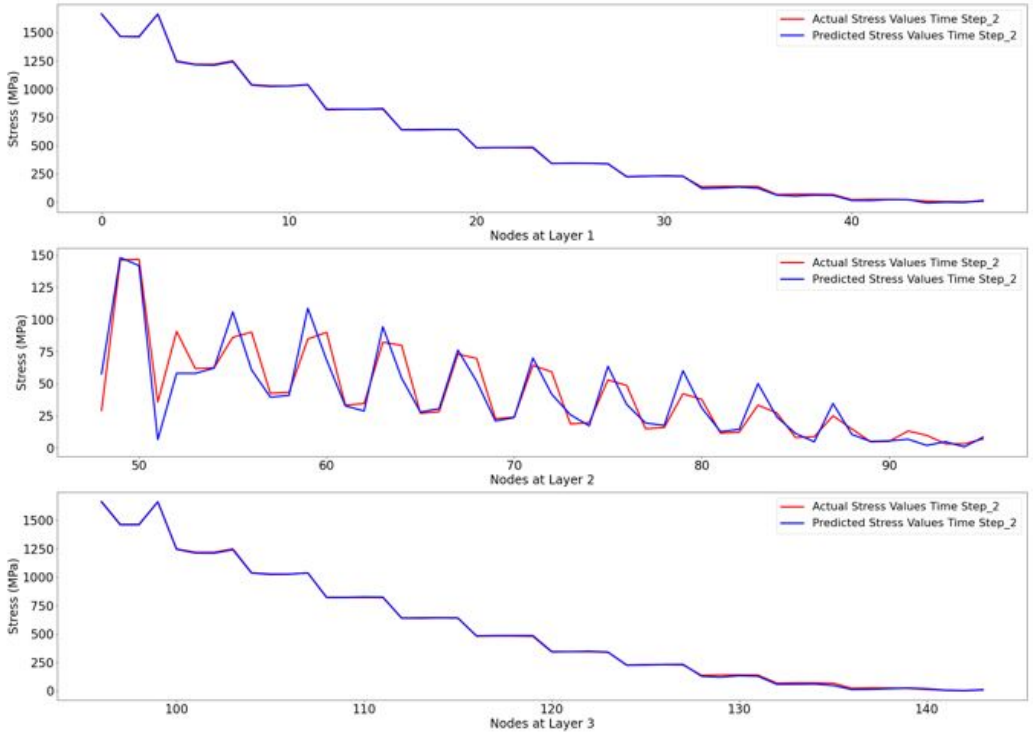

FIGURE 6.27: Nodal stresses for time step 2

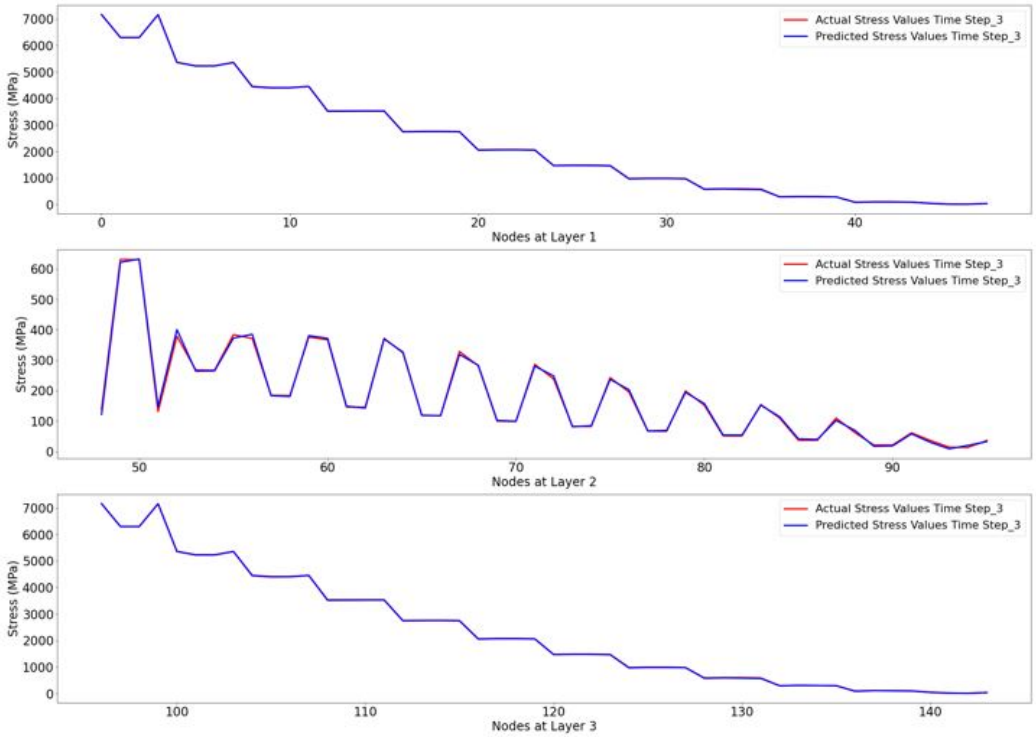

FIGURE 6.28: Nodal stresses for time step 3

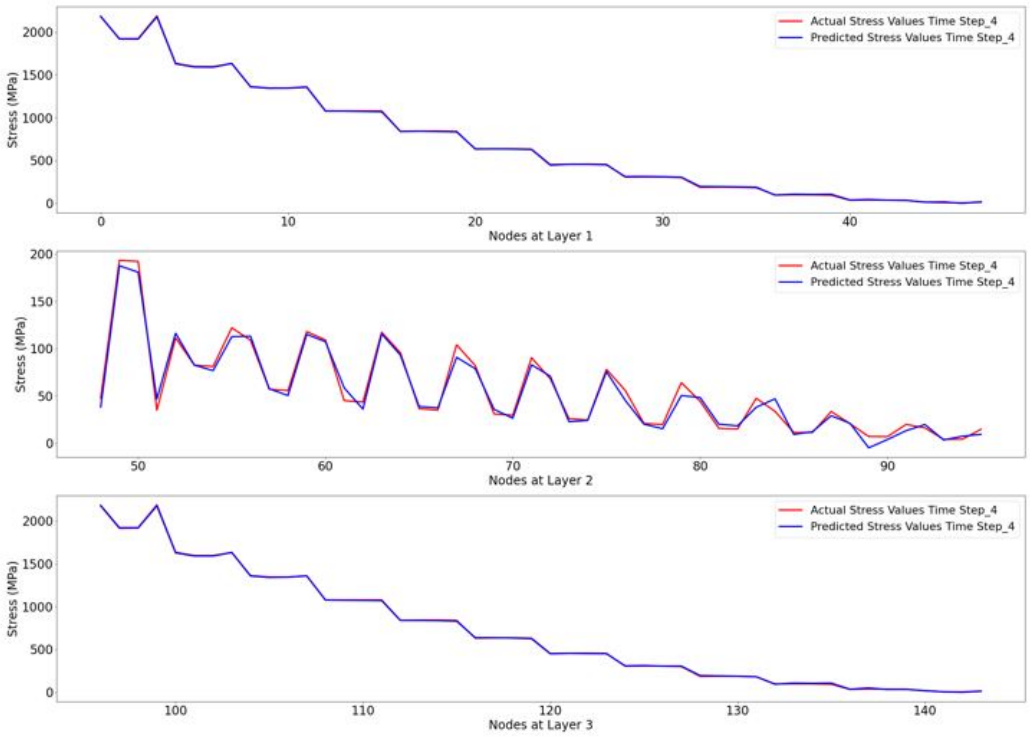

FIGURE 6.29: Nodal stresses for time step 4

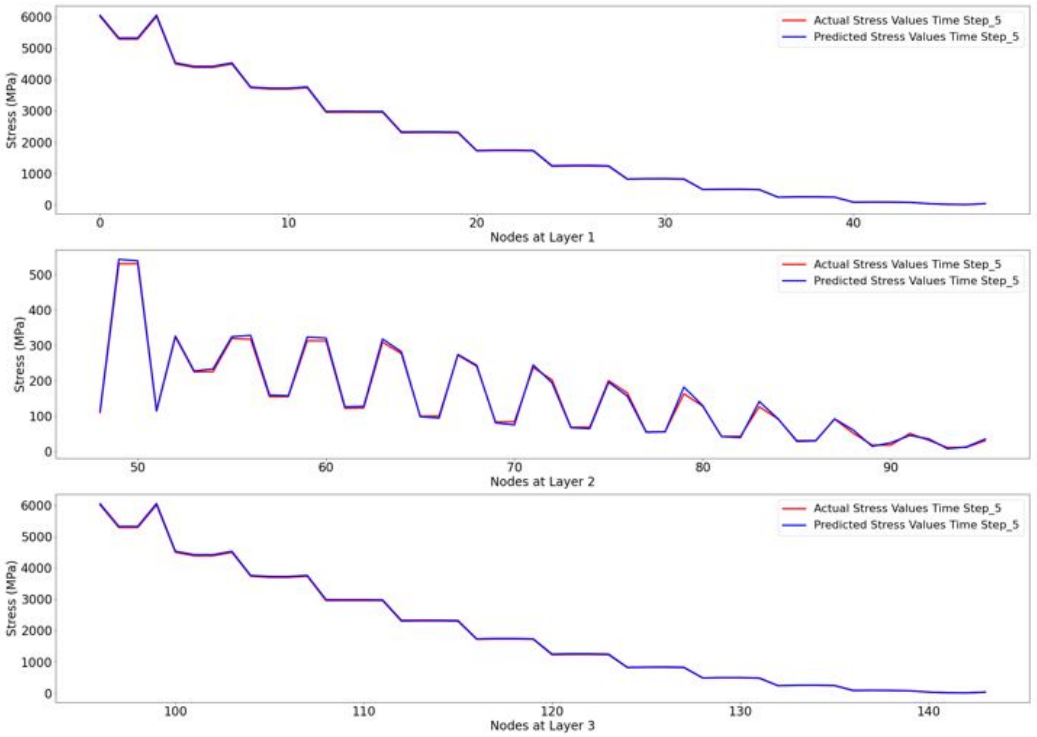

FIGURE 6.30: Nodal stresses for time step 5

Plots in figure 6.26 to 6.30 show the actual and predicted values for stress at nodal points on different layers of cantilever beam. The overlapping nature of lines indicate good stress value predictions.

Stress Error Visualization

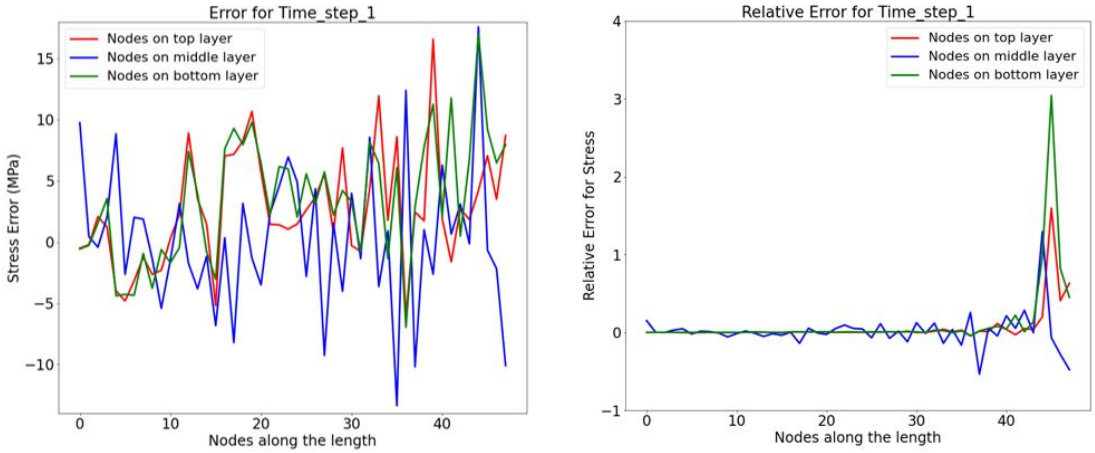

FIGURE 6.31: Error and relative error for stress at time step 1

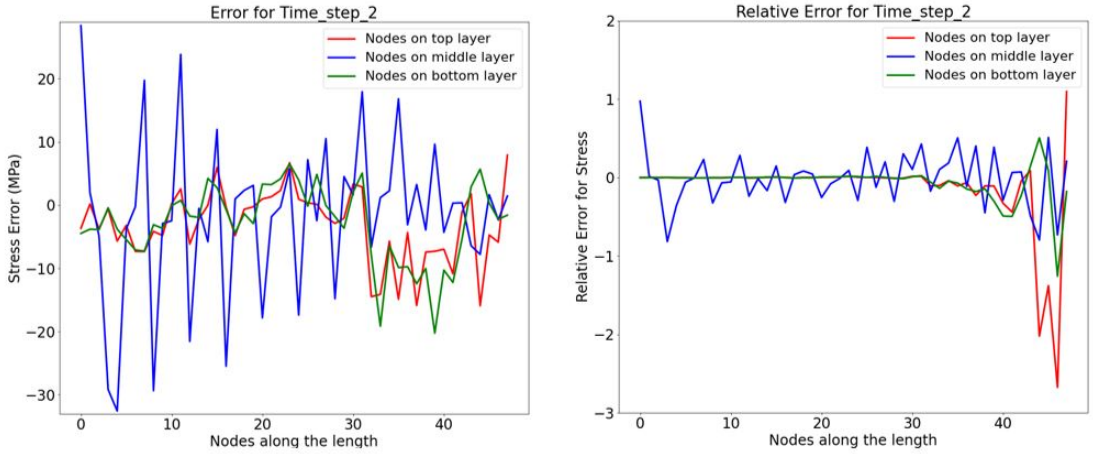

FIGURE 6.32: Error and relative error for stress at time step 2

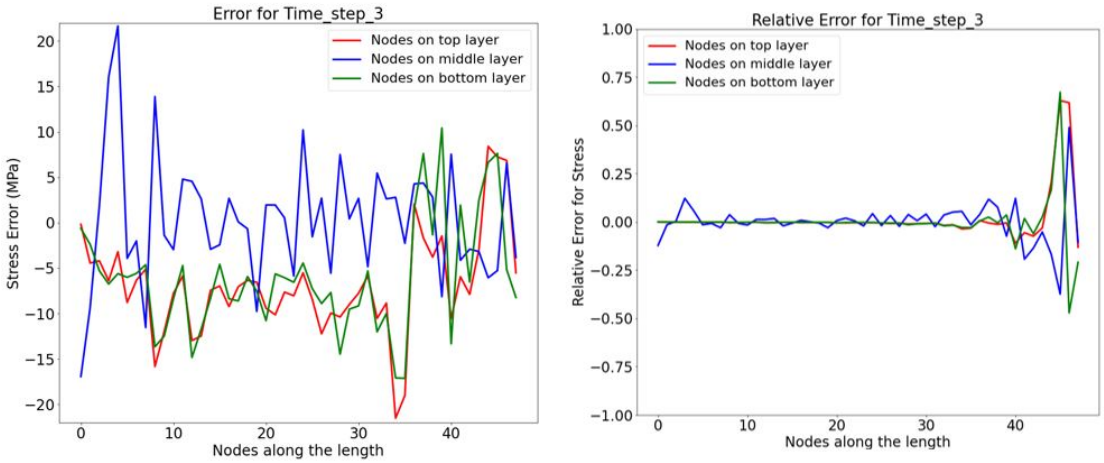

FIGURE 6.33: Error and relative error for stress at time step 3

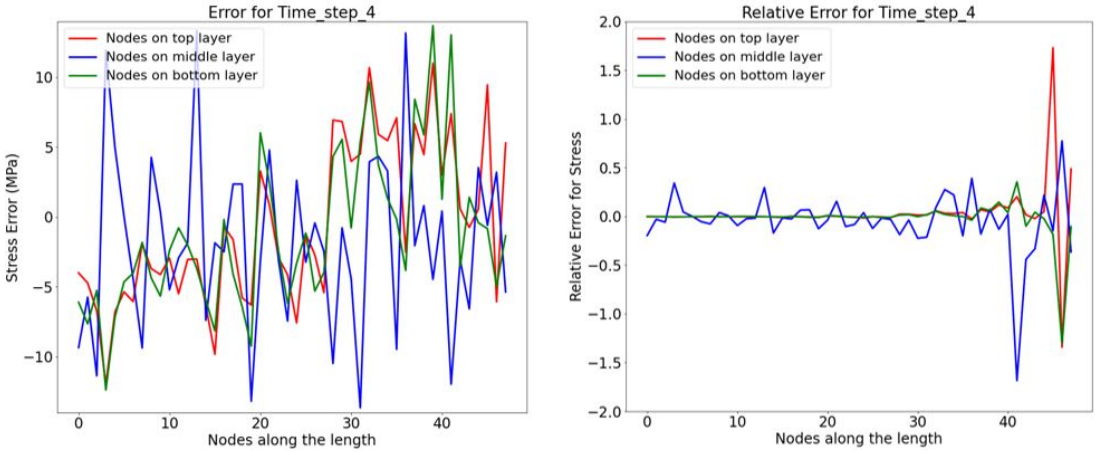

FIGURE 6.34: Error and relative error for stress at time step 4

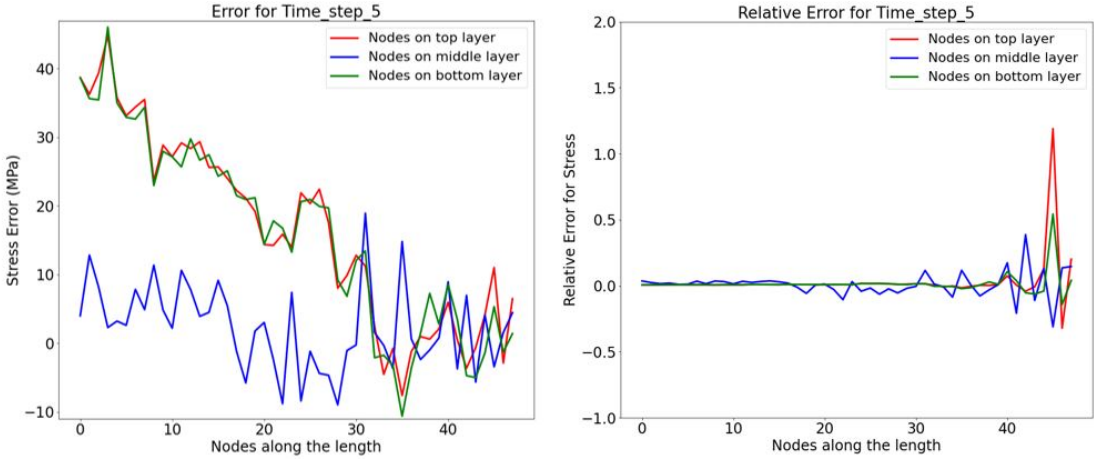

FIGURE 6.35: Error and relative error for stress at time step 5

Figure 6.31 to 6.35 show the error and relative error plots for stress at five sequential time steps. The observed spike in the relative error plots is because of stress values close to zero, making the relative error high. The maximum stress error of around 50 MPa was observed in time step 5.

6.1.2 Analysis with Gated Graph Architecture for Coarse Mesh

The Gated Graph Neural Network (GG-NN) architecture from [56] is an extension of work from [58] on graph neural networks. It uses gated recurrent unit which are more suitable when output is some form of a sequence. Considering the analysis of cantilever beam for different time steps where the input node feature is sequence of forces that would produce corresponding output values in for of sequence of deformation and stress values, this architecture would thus be suitable for our application. The propagation model follows the following system of equations,

$$h_v^{(1)} = [x_v^T, 0]^T \tag{6.2}$$

$$a_v^{(t)} = A_{v:}^T [h_1^{(t-1)T} \ldots h_{|v|}^{(t-1)T}]^T + b \tag{6.3}$$

$$z_v^t = \sigma(W^z a_v^{(t)} + U^z h_v^{(t-1)}) \tag{6.4}$$

$$r_v^t = \sigma(W^r a_v^{(t)} + U^r h_v^{(t-1)}) \tag{6.5}$$

$$\widetilde{h_v^{(t)}} = \tanh(W a_v^{(t)} + U(r_v^t \odot h_v^{(t-1)}) \tag{6.6}$$

$$h_v^{(t)} = (1 - z_v^t) \odot h_v^{(t-1)} + z_v^t \odot \widetilde{h_v^{(t)}} \tag{6.7}$$

The first equation is the initialization step which sets the node annotations for first hidden state with zero padding. The second equation step passes the information between the nodes. The rest equations are GRU-like updates with z and r being the update and reset gate respectively. σ is the logistiic regression and $\odot$ is element-wise multiplication. We follow similar structure to that of GCN with 3 layered gated graph combined with 2 layers of MLP. The architecture is depicted in figure below.

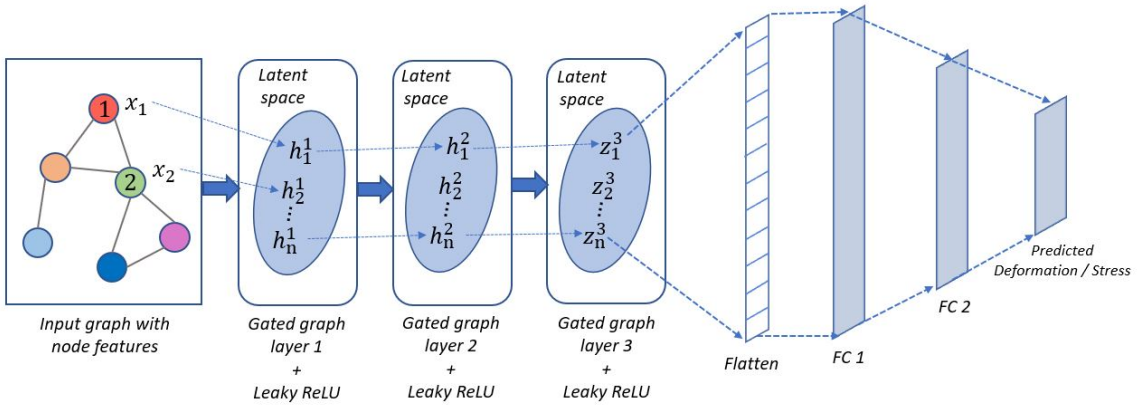

FIGURE 6.36: Model with gated graph architecture

The results for the deployed DNN model with GG-NN architecture are as follows.

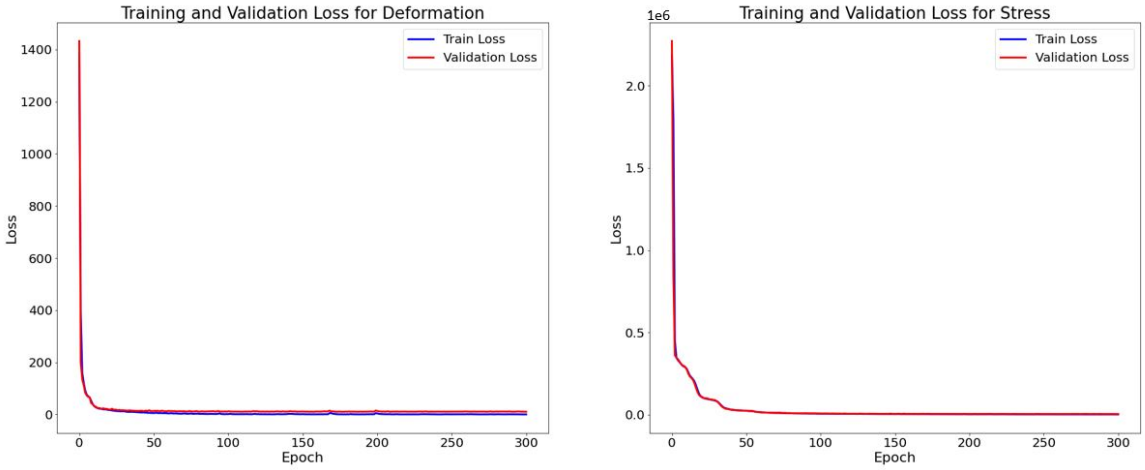

FIGURE 6.37: Training and validation curve for model with GG-NN architecture

3D Plots for Deformation

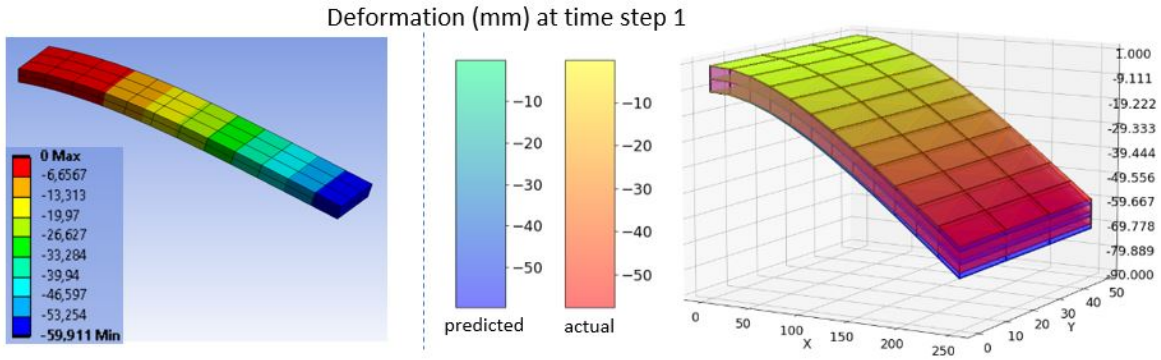

FIGURE 6.38: FEM vs. GG-NN result for deformation at time step 1

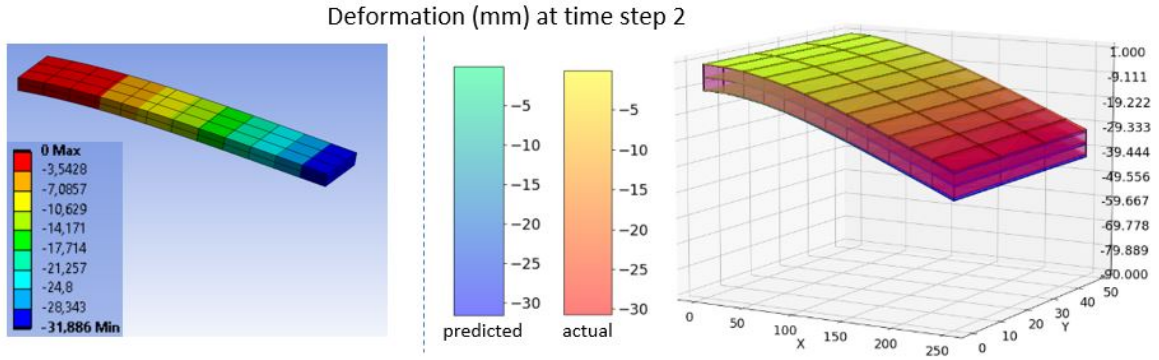

FIGURE 6.39: FEM vs. GG-NN result for deformation at time step 2

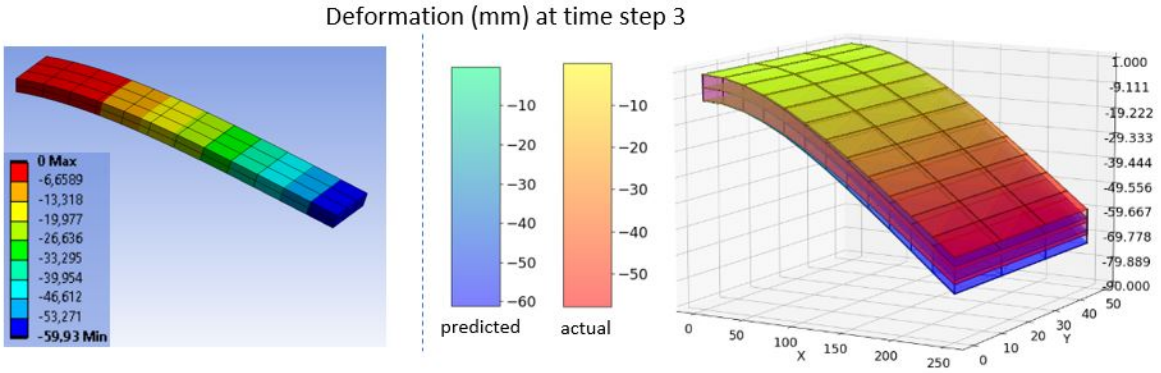

FIGURE 6.40: FEM vs. GG-NN result for deformation at time step 3

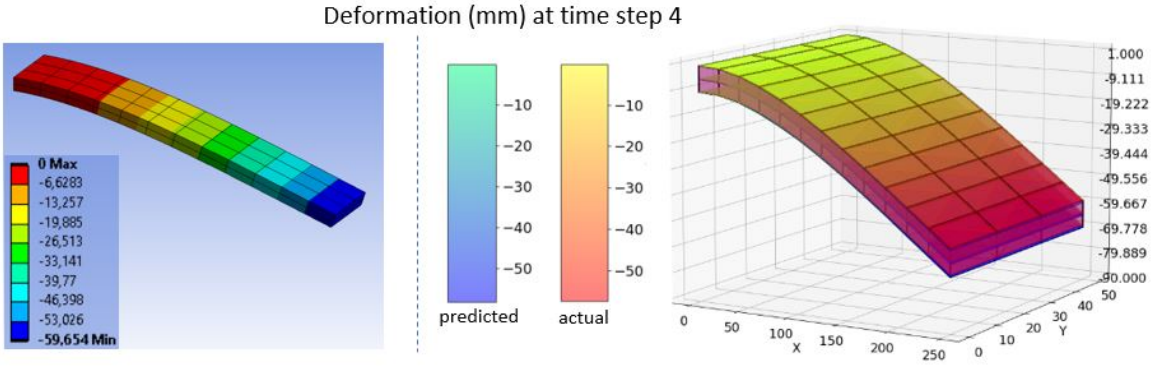

FIGURE 6.41: FEM vs. GG-NN result for deformation at time step 4

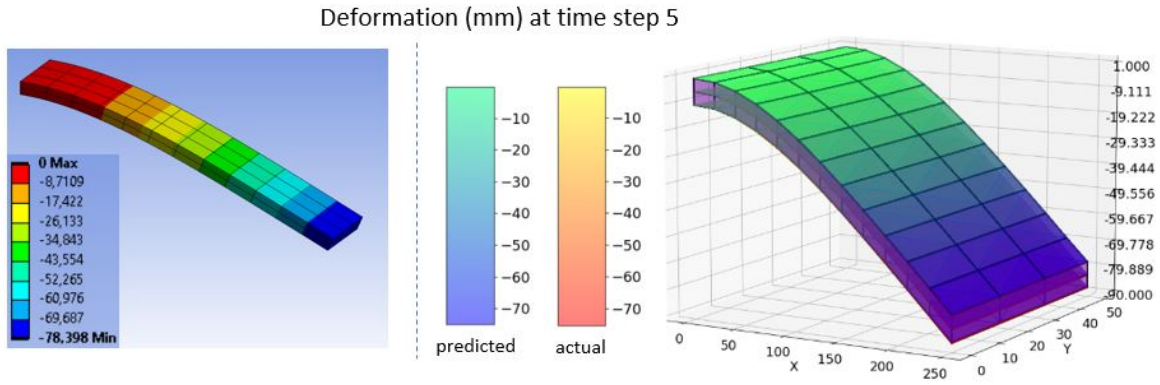

FIGURE 6.42: FEM vs. GG-NN result for deformation at time step 5

From figure 6.38 to 6.42 it stated that the deformation predictions are good since the beams overlap in all time steps except in time step 3 where a slight separation can be seen at the free end of cantilever.

The detailed plots for nodal deformation can be seen in Appendix section A.1.

Deformation Error Visualization

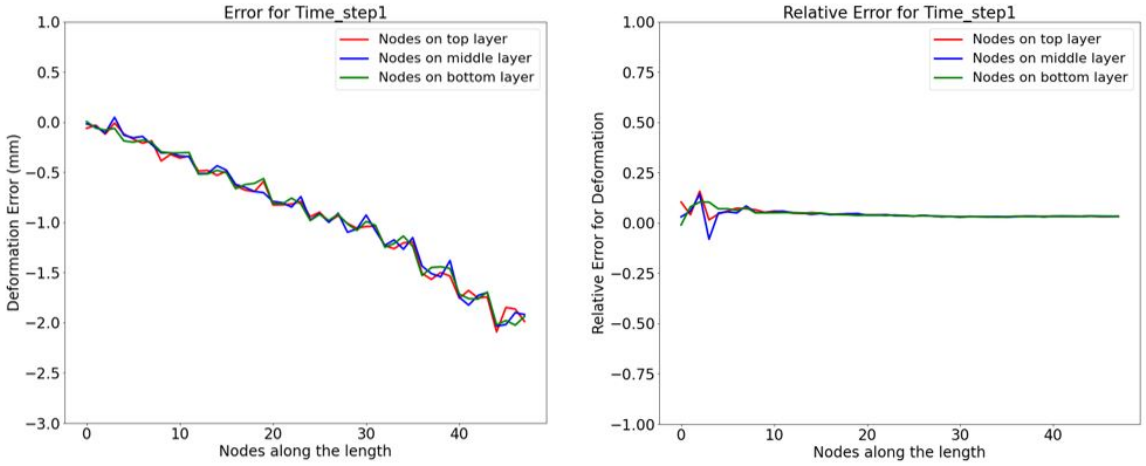

FIGURE 6.43: Error and relative error for deformation at time step 1

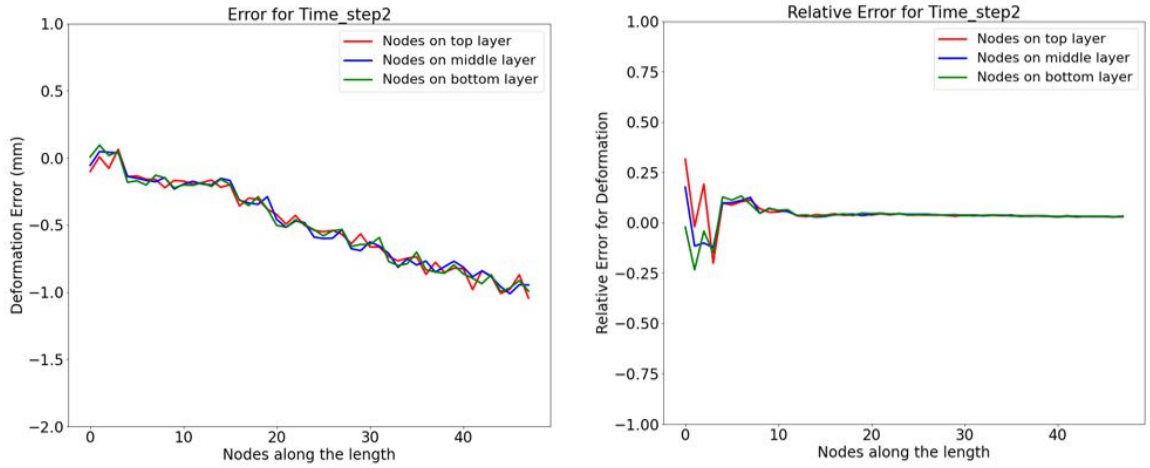

FIGURE 6.44: Error and relative error for deformation at time step 2

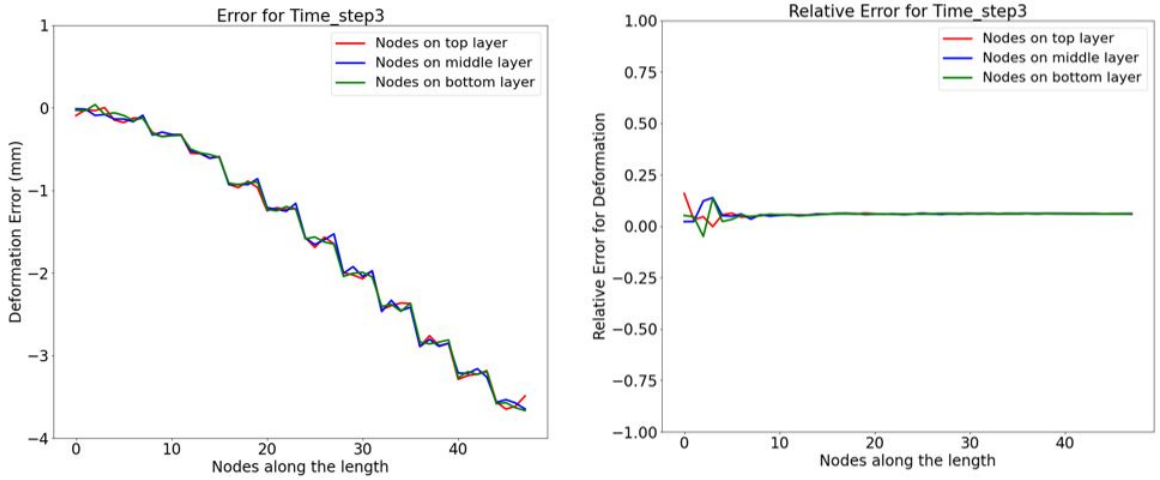

FIGURE 6.45: Error and relative error for deformation at time step 3

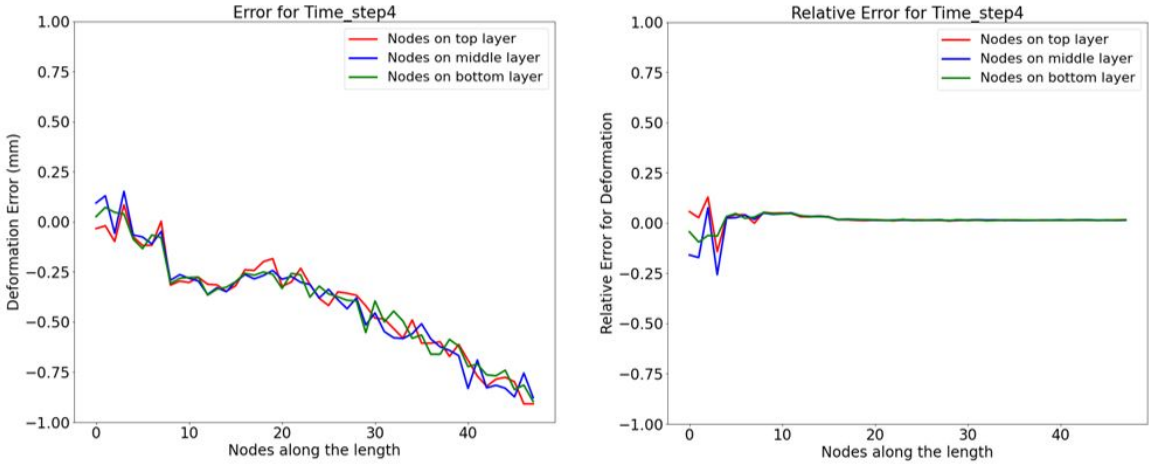

FIGURE 6.46: Error and relative error for deformation at time step 4

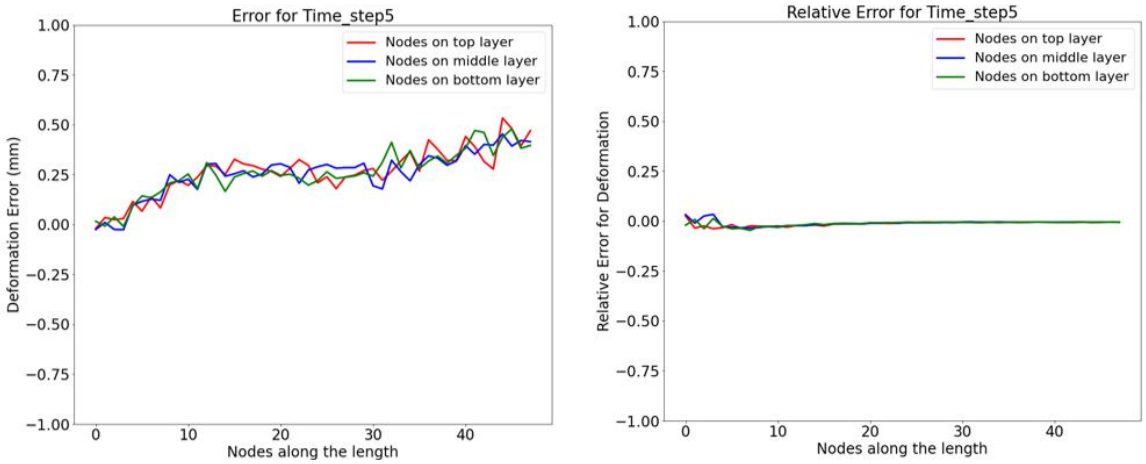

FIGURE 6.47: Error and relative error for deformation at time step 5

From figure 6.43 to 6.47 it can be seen that the relative error is close to zero even for small deformation values near fixed end of cantilever. This shows comparatively better prediction than GCN for small values of deformation. Also, maximum deformation error close to 4mm was seen in time step 3. The error plots thus indicate good performance of GG-NN model on deformation predictions.

3D Plots for Stress

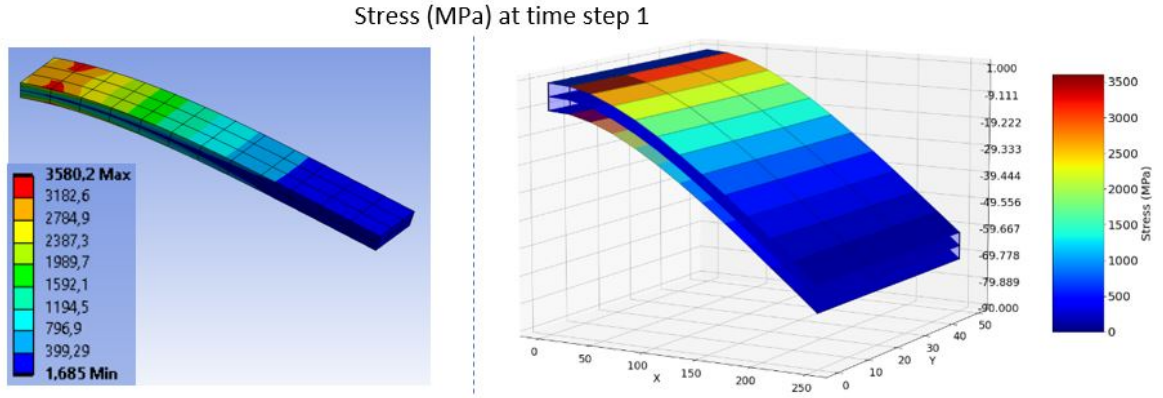

FIGURE 6.48: FEM vs. GG-NN result for stress at time step 1

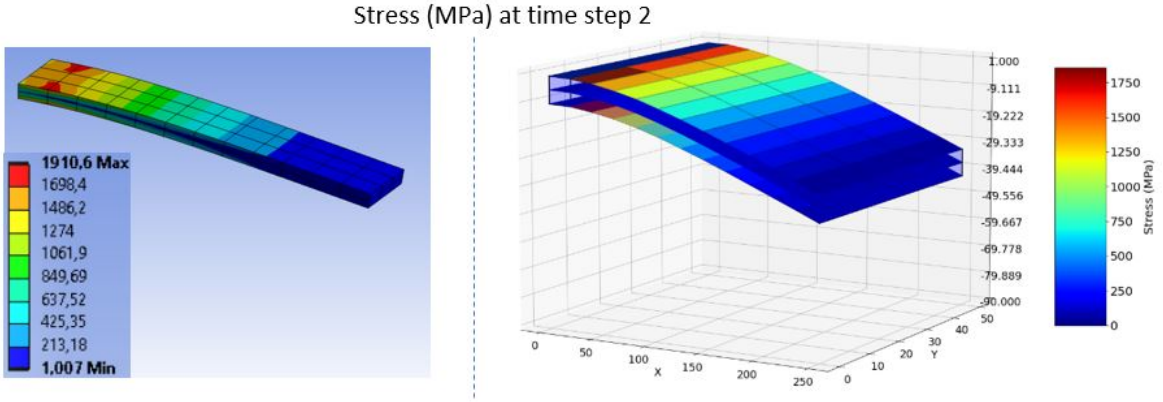

FIGURE 6.49: FEM vs. GG-NN result for stress at time step 2

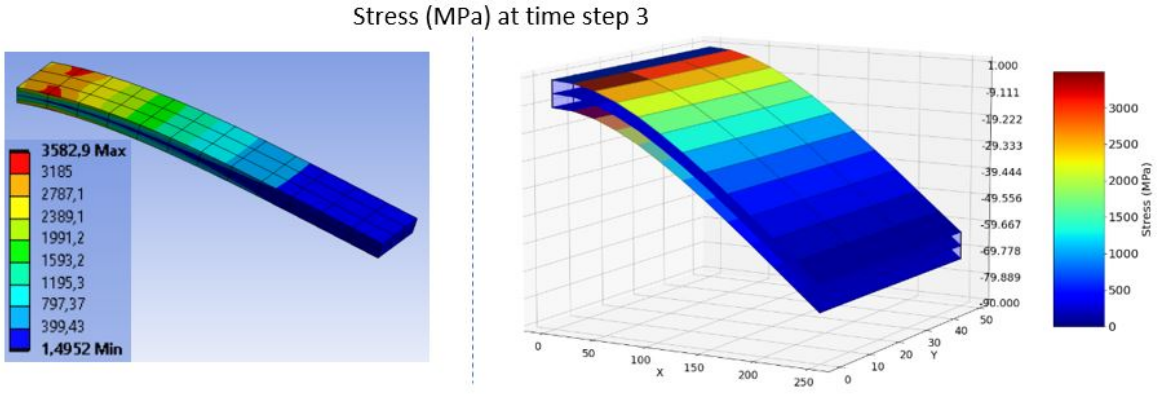

FIGURE 6.50: FEM vs. GG-NN result for stress at time step 3

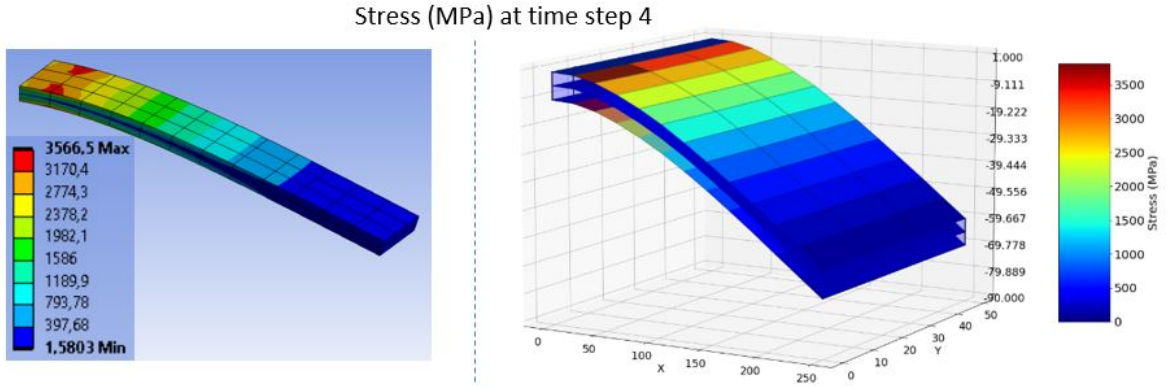

FIGURE 6.51: FEM vs. GG-NN result for stress at time step 4

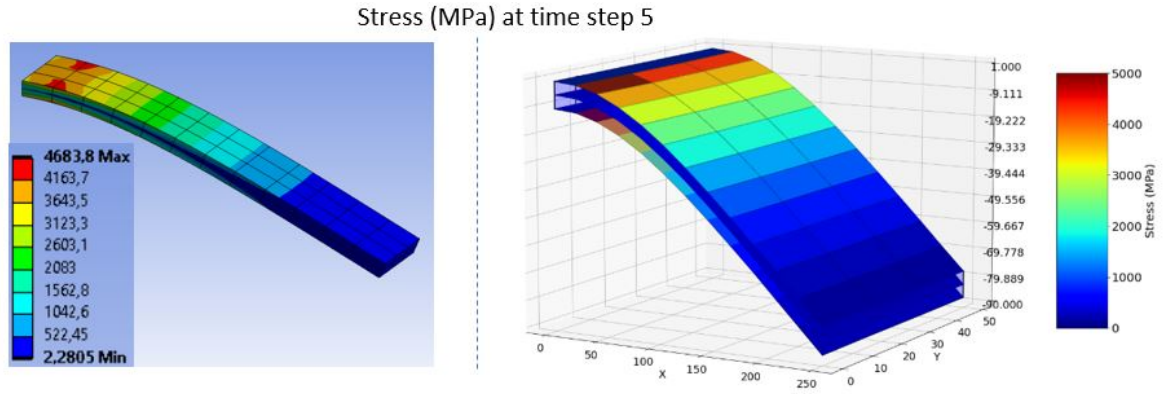

FIGURE 6.52: FEM vs. GG-NN result for stress at time step 5

The stress plots from figure 6.48 to 6.52 indicate that the GG-NN model successfully identifies the different stress intensities in different regions of cantilever beam. Also, by comparing the values on the colorbar it can be seen that there is very less difference in actual FEA result and predicted model result.
The plots for actual nodal stress values can be seen in Appendix section A.1

Stress Error Visualization

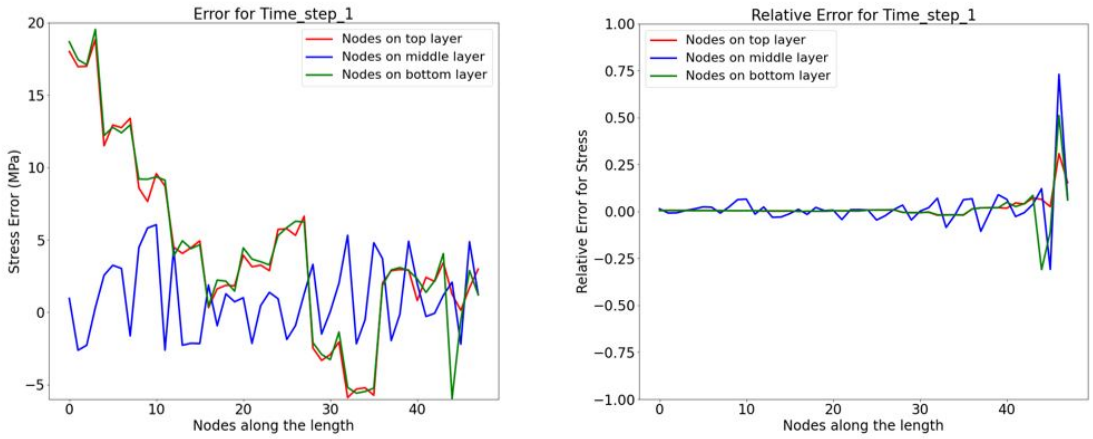

FIGURE 6.53: Error and relative error for stress at time step 1

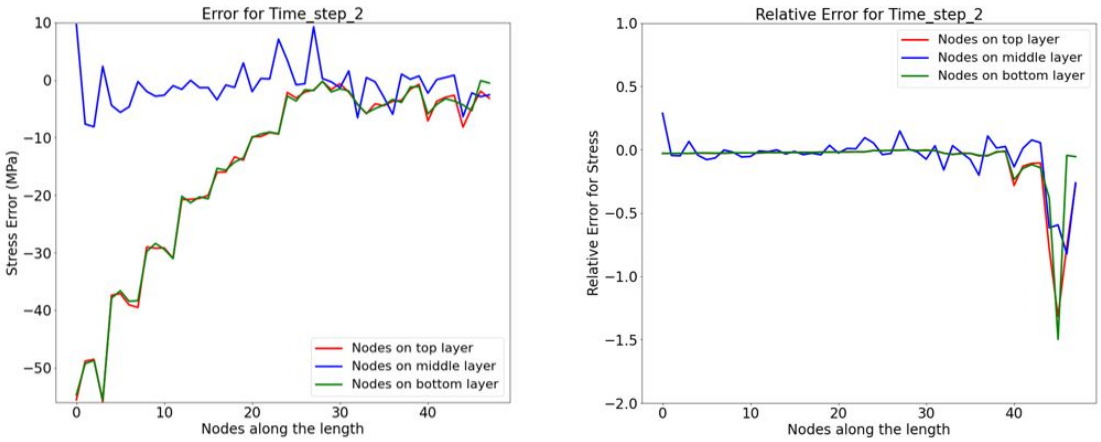

FIGURE 6.54: Error and relative error for stress at time step 2

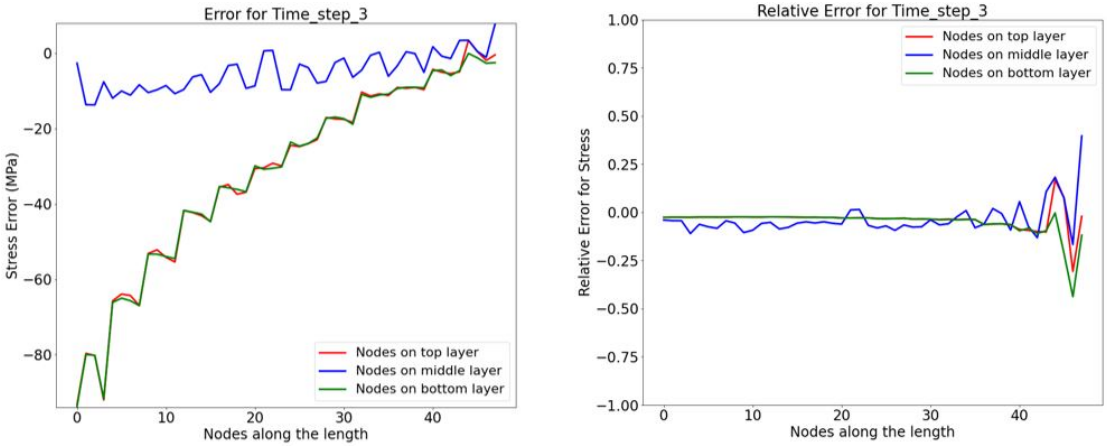

FIGURE 6.55: Error and relative error for stress at time step 3

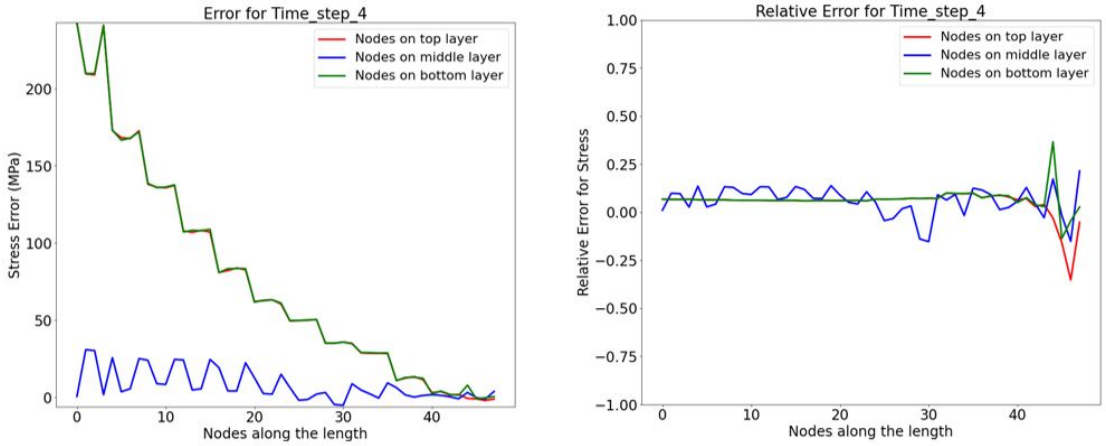

FIGURE 6.56: Error and relative error for stress at time step 4

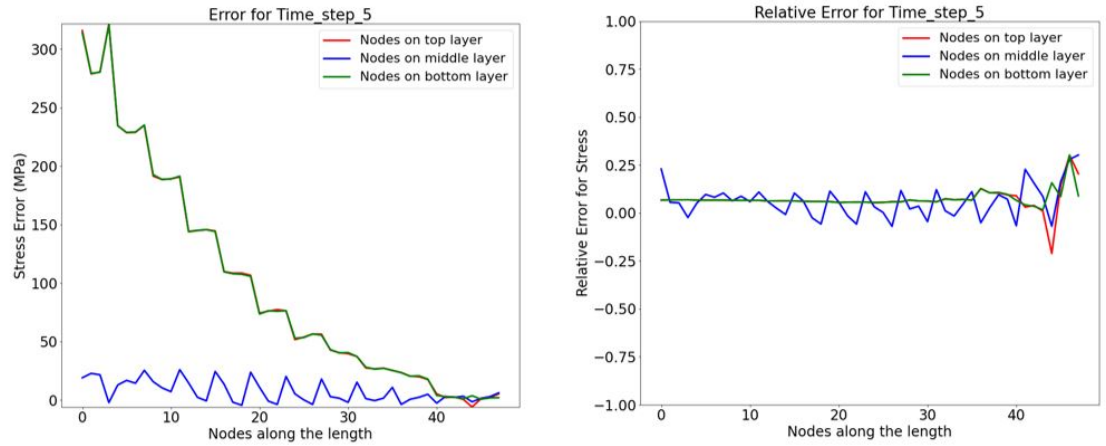

FIGURE 6.57: Error and relative error for stress at time step 5

The stress error and relative stress error plots from figure 6.53 to 6.57 show high difference between actual and predicted values for time step 4 and 5. Although the difference is high, its significance with respect to actual value is low as seen from the relative error plots.

6.1.3 Effect of Shared Network vs. Independent Network

Since the output space contains two types of predictions viz. deformation and stress, two different methods were experimented using architectures shown in figure 6.58.

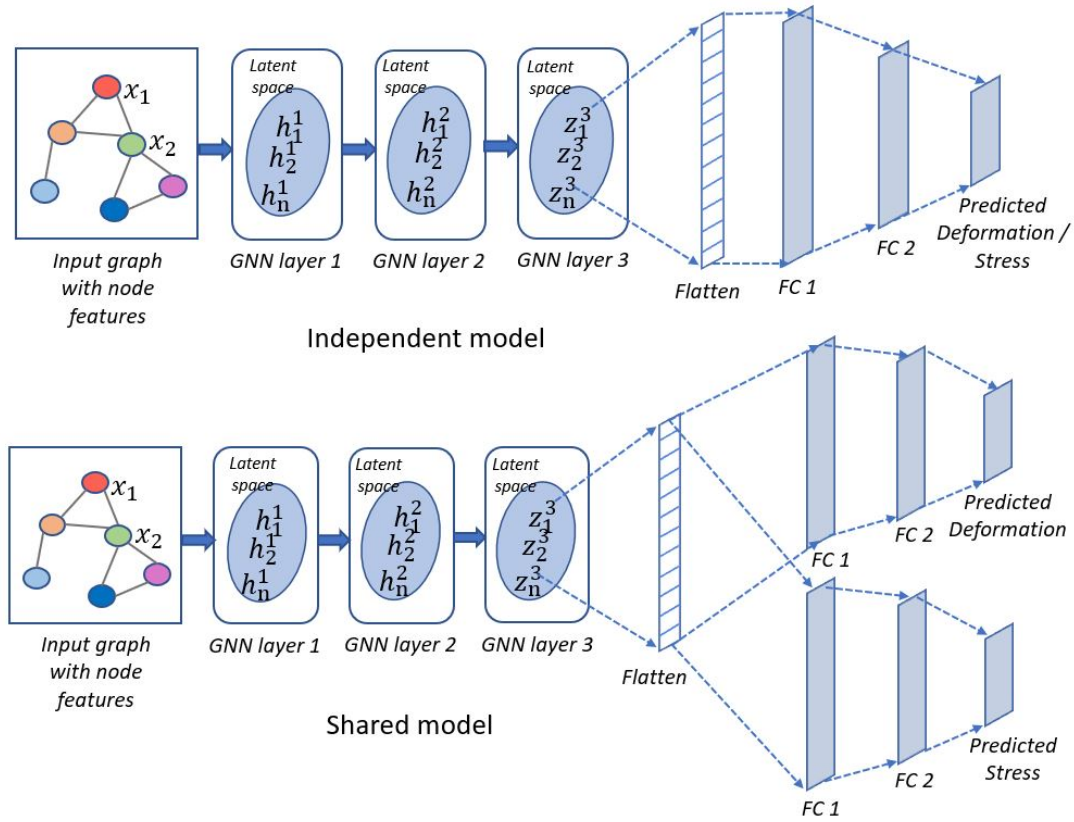

FIGURE 6.58: Independent network and shared network architectures

First, the predictions were done individually using independent network architecture for deformation and stress. In second method, the graph embedded data was shared by two MLPs each predicting deformation and stress respectively. While doing so it was necessary to scale down the stress loss and express the total loss as linear combination of both losses to avoid imbalance in updation of weights during backpropagation [59]. It was found that, although the computation time for individual predictions from independent networks was double to that of shared network, it gave a slight advantage of more accurate predictions. The independent network architecture gave MAE of 4.0248 while shared network gave 5.0514 MAE value for coarse mesh case. Similar trend was observed for the other two datasets as well. Hence, the independent network was considered as the go-to model.

6.1.4 Varying the Number of Layers in Neural Network

The effect of number of GNN layers and MLP layers on the performance of model has been summarized in table 6.1.

Variation	GNN layers	MLP layers	Mean Absolute Error
1	3	2	4.0248
2	4	2	6.3214
3	5	2	5.5788
4	6	2	10.3
5	7	2	10.005
6	3	3	11.3667
7	3	4	87.89
8	3	5	223.2431
9	3	6	165.9339

TABLE 6.1: Effect of number of layers on the prediction quality for coarse mesh dataset

The general convention is that with increase in layers of neural network the quality of prediction improves. However, it was observed in our case that the network gave good results for shallow architecture. After a limit of 5 GNN layers and 3 MLP layers, the model performance deteriorates and it fails to generalize the predictions giving high MAE. Similar behaviour was observed for the other two datasets as well. A shallow network with 3 graph network layers and 2 MLP layers gave the best result.

6.1.5 Effect of Different Activation Functions

The main purpose of activation function is to introduce non-linearity and make the network learn better the relationship between input and output. Hence, proper selection of activation function has high impact on the performance of a model. Table 6.2 and 6.3 summarizes the effect of various activation functions. It was observed that the combination of Leaky ReLU activation function in GNN and MLP layers gave the best result.

Activation Function	Mean Absolute Error
ReLU	4.2082
Leaky ReLU	4.0248
Tanh	157.0269
Softplus	7.4891

TABLE 6.2: Effect of activation function in MLP layers on the model performance for coarse mesh dataset

Activation Function	Mean Absolute Error
ReLU	4.4131
Leaky ReLU	4.0284
Tanh	88.2447
Softplus	7.0549

TABLE 6.3: Effect of activation functions in GNN layers on the model performance for coarse mesh dataset

6.2 Beam with Finer Mesh

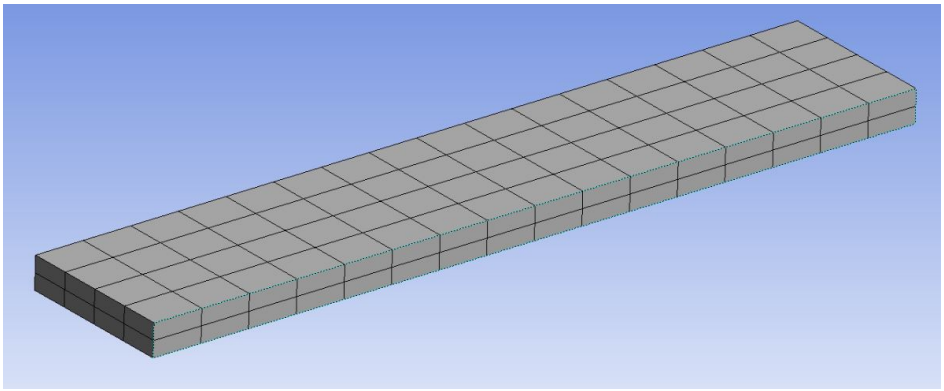

FIGURE 6.59: Cantilever beam with finer meshing

The FEA results obtained from coarse mesh setting give a less refined result enough to identify critical areas for failure or improvement in design or boundary conditions. After having identified the potential risks and critical regions, it is essential to converge the results into more reliable and accurate values. This is done by increasing the number of elements and nodes or in other words- creating a finer mesh. The general convention is to obtain more accurate and reliable results at the ares of interest by increasing the elements locally. However, in our case of cantilever in order to generalize, we create an overall finer mesh without restricting to a local zone. The finer mesh setting of cantilever beam is shown in figure below with 128 elements and 255 nodes.

6.2.1 Analysis with GCN Architecture for Finer Mesh

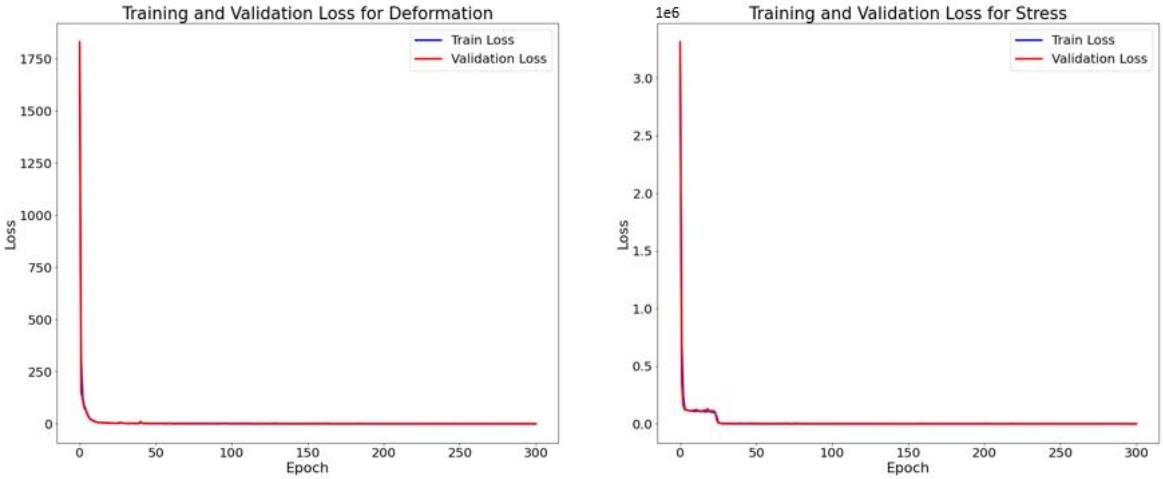

FIGURE 6.60: Training and validation curve for model with GCN architecture

3D Plots for Deformation

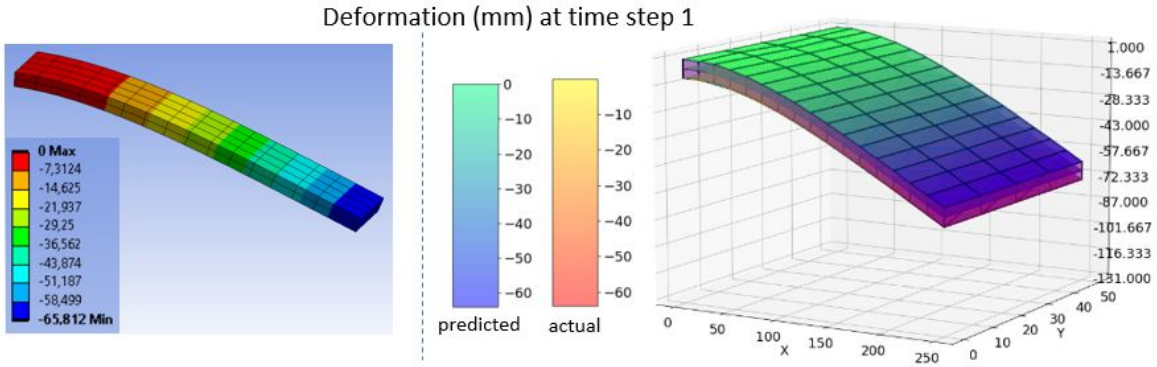

FIGURE 6.61: FEM vs. GCN result for deformation at time step 1

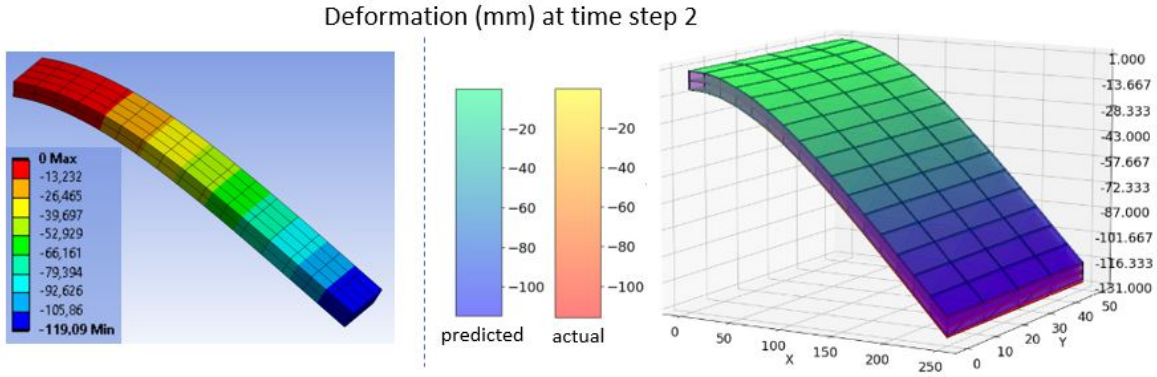

FIGURE 6.62: FEM vs. GCN result for deformation at time step 2

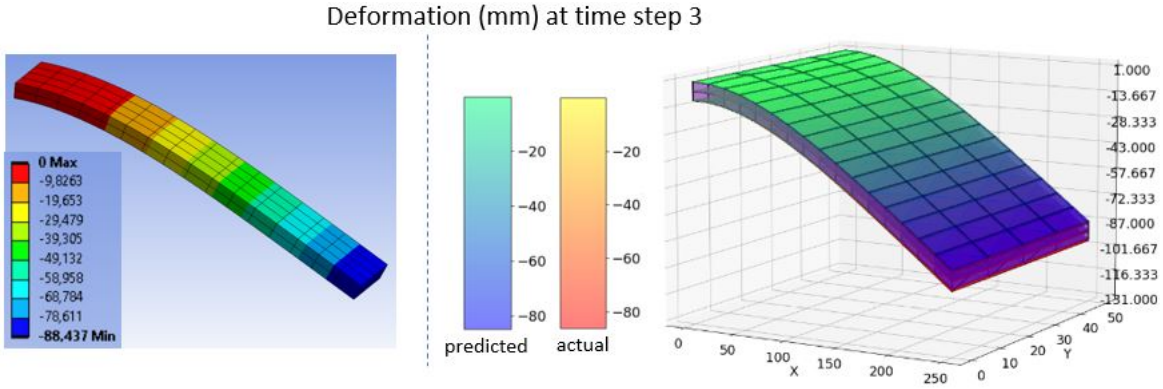

FIGURE 6.63: FEM vs. GCN result for deformation at time step 3

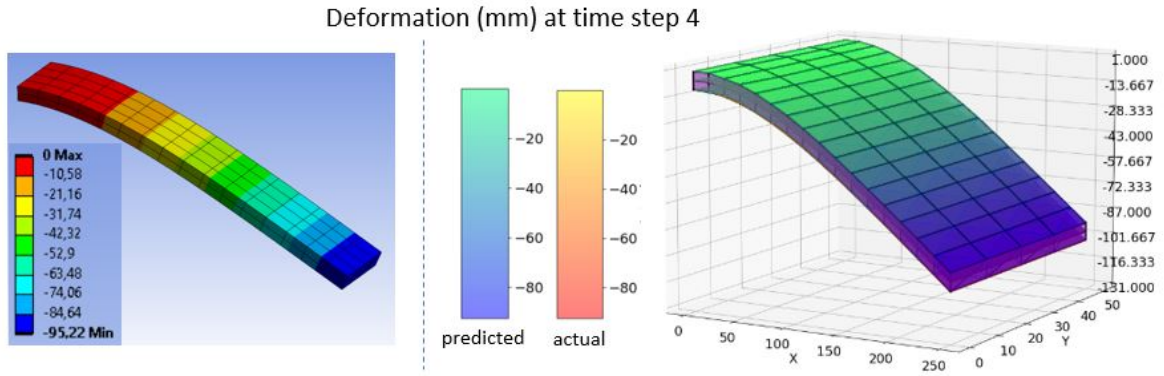

FIGURE 6.64: FEM vs. GCN result for deformation at time step 4

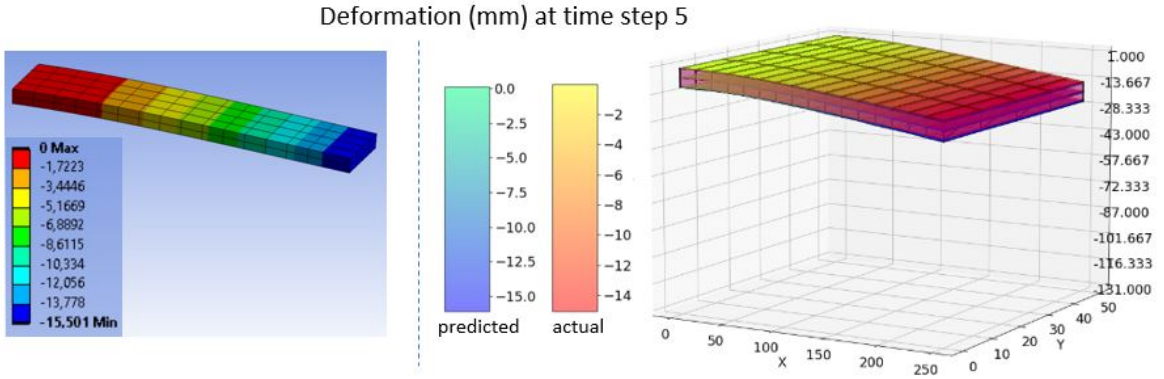

FIGURE 6.65: FEM vs. GCN result for deformation at time step 5

From the 3D plots in figure 6.61 to 6.70 it can be concluded that the GCN architecture gave excellent results with the deformation values identical to that obtained from the FEA software for all the time steps.

The plots for actual nodal deformation values can be refered in Appendix section A.2.

Deformation Error Visualization

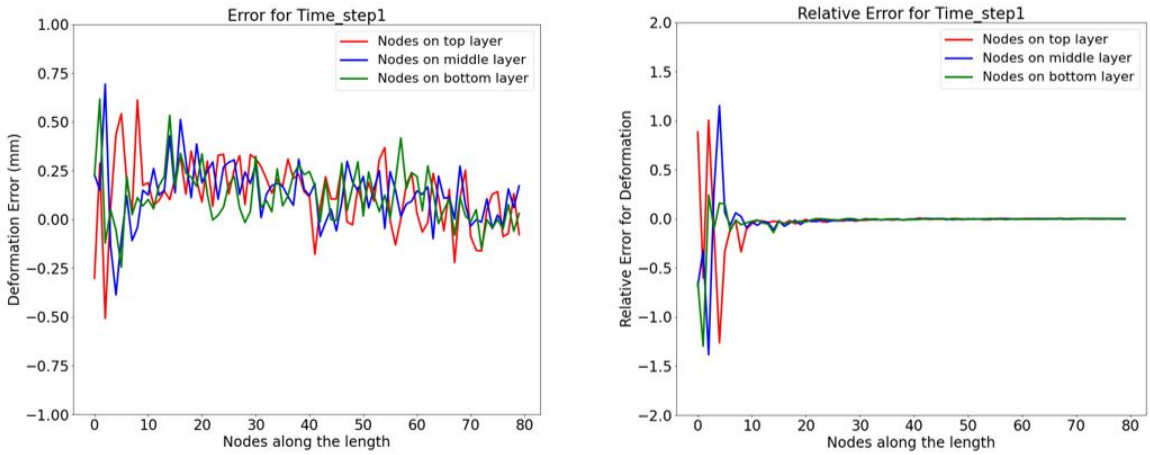

FIGURE 6.66: Error and relative error for deformation at time step 1

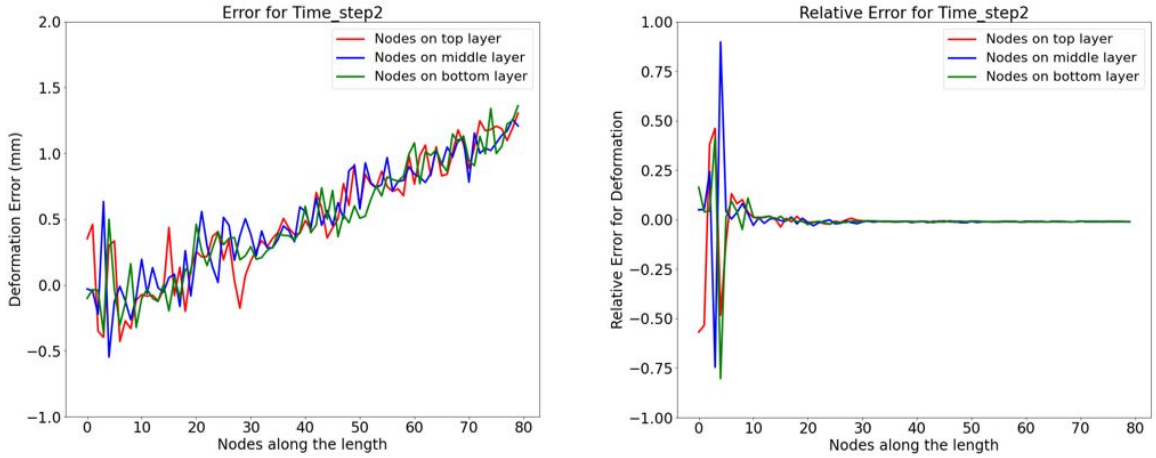

FIGURE 6.67: Error and relative error for deformation at time step 2

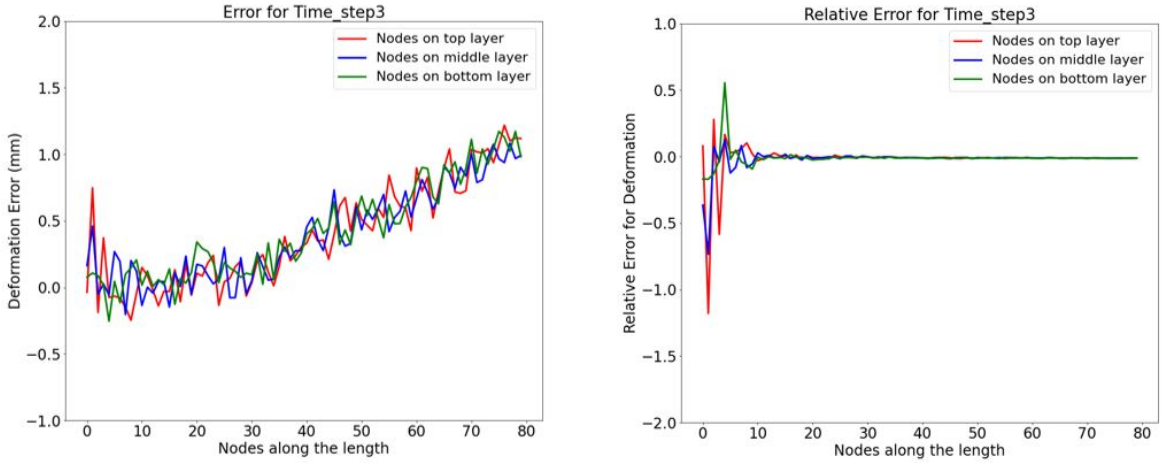

FIGURE 6.68: Error and relative error for deformation at time step 3

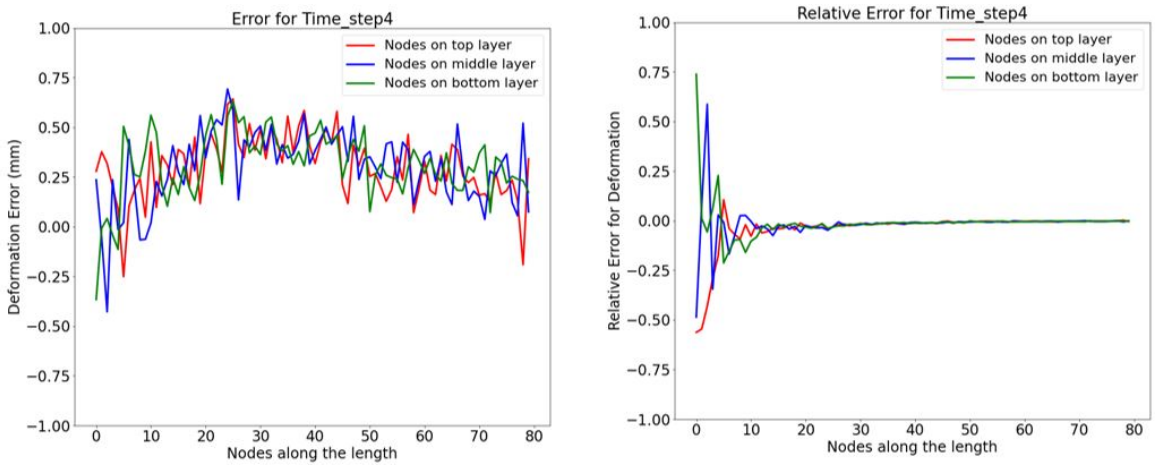

FIGURE 6.69: Error and relative error for deformation at time step 4

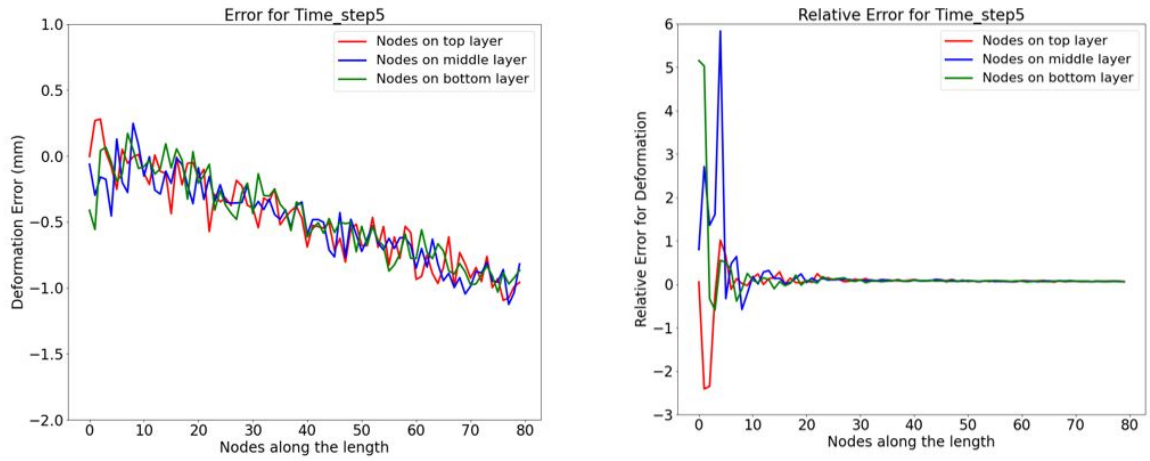

FIGURE 6.70: Error and relative error for deformation at time step 5

The error plots in figure 6.66 to 6.70 confirm the excellent prediction of deformation values with error below 1mm. The relative error plots show slight fluctuations for prediction of small deformation values close to zero.

3D Plots for Stress

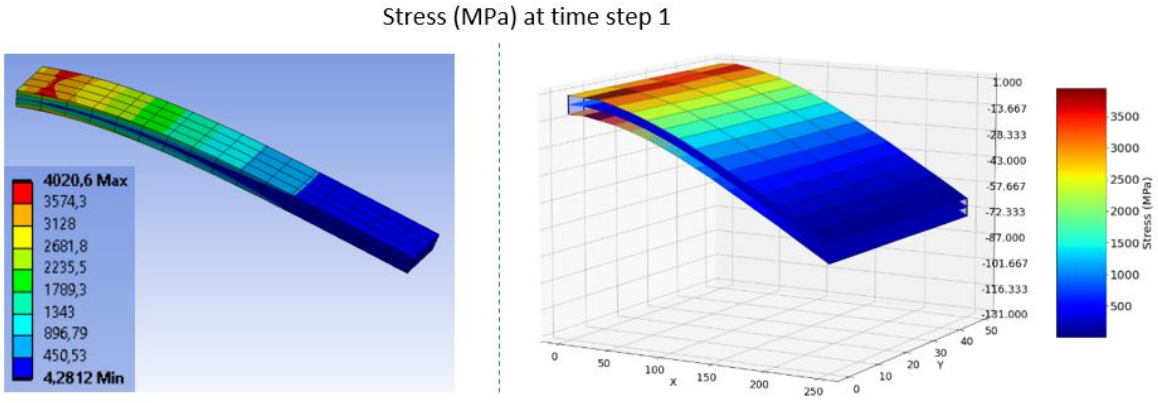

FIGURE 6.71: FEM vs. GCN result for stress at time step 1

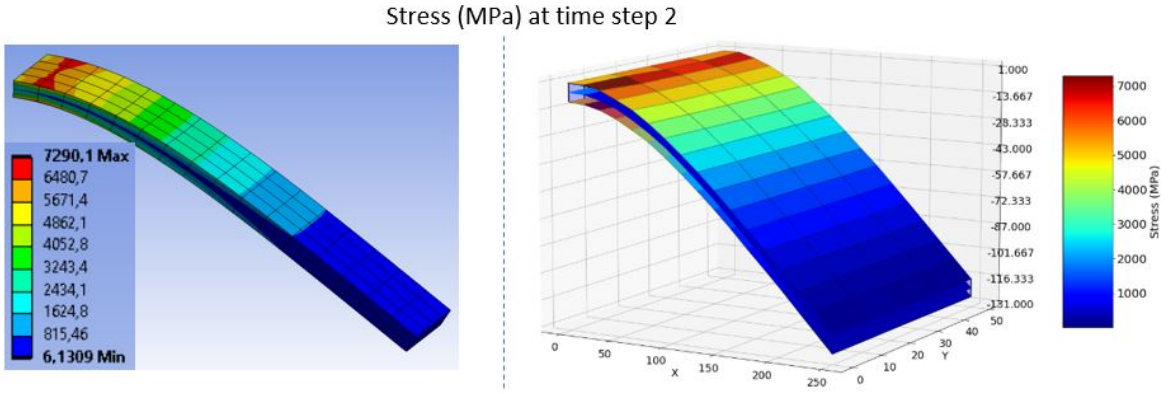

FIGURE 6.72: FEM vs. GCN result for stress at time step 2

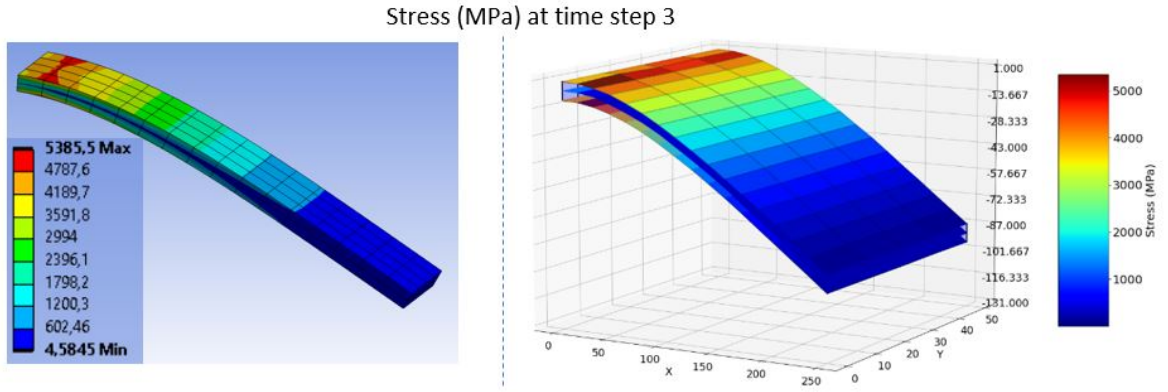

FIGURE 6.73: FEM vs. GCN result for stress at time step 3

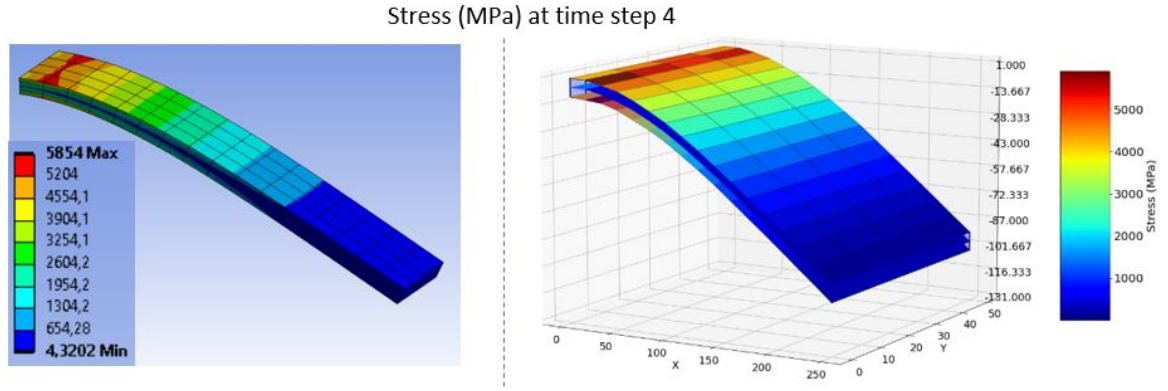

FIGURE 6.74: FEM vs. GCN result for stress at time step 4

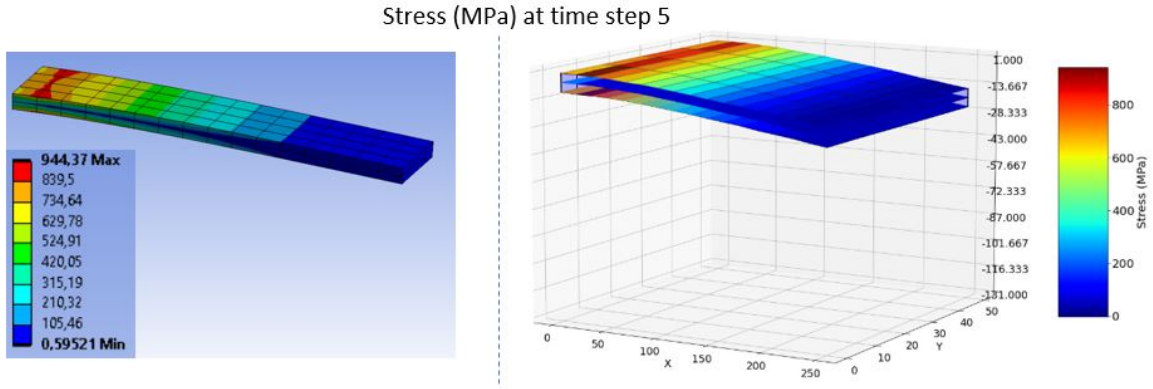

FIGURE 6.75: FEM vs. GCN result for stress at time step 5

The 3D stress plots shown in figure 6.71 to 6.75 indicate that the GCN model is successful in predicting the stress distribution in different regions. Also the similarity in actual FEA value and model predicted values on the colorbars imply good accuracy.

The actual and predicted nodal stress values can be visualized in plots from Appendix section A.2.

Stress Error Visualization

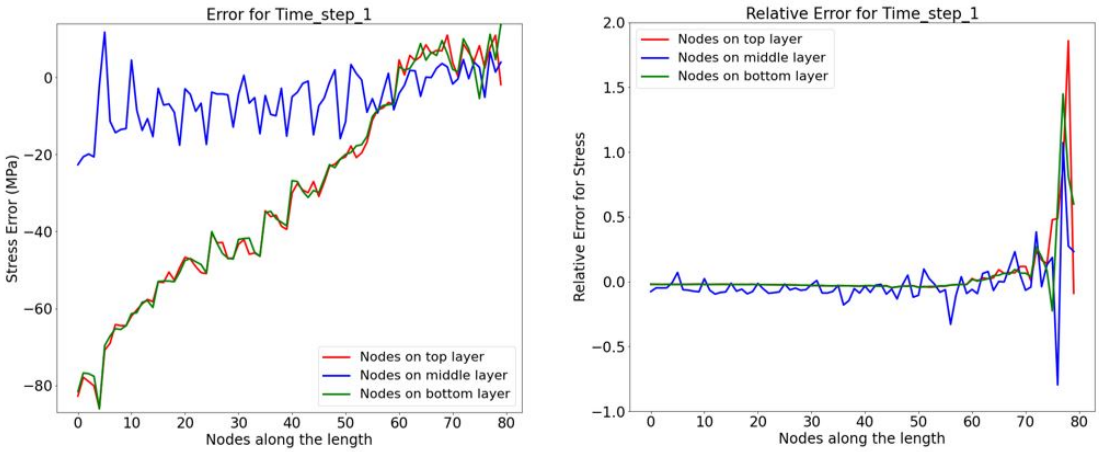

FIGURE 6.76: Error and relative error for stress at time step 1

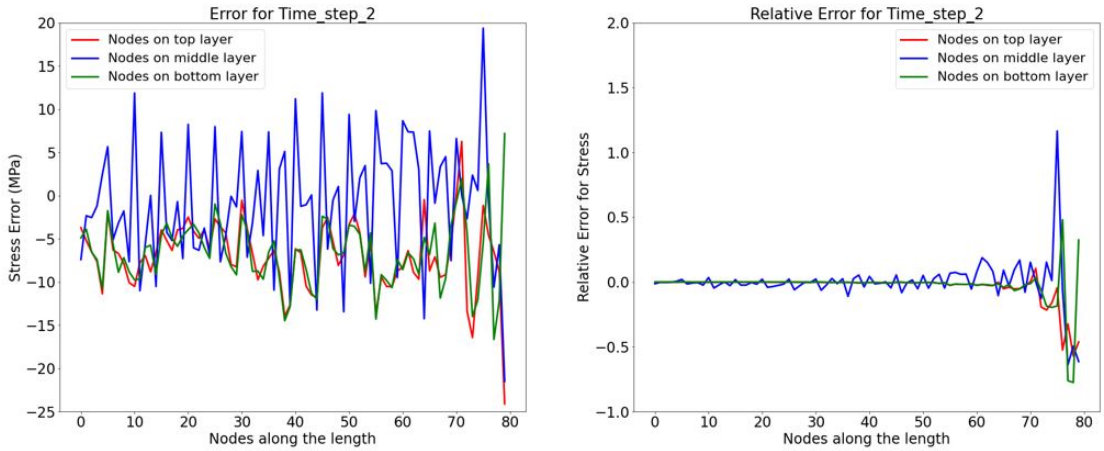

FIGURE 6.77: Error and relative error for stress at time step 2

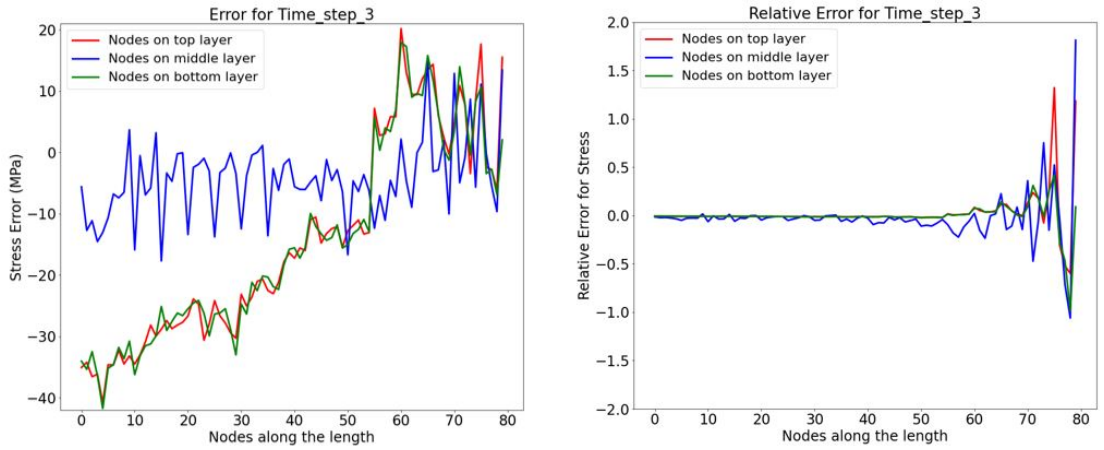

FIGURE 6.78: Error and relative error for stress at time step 3

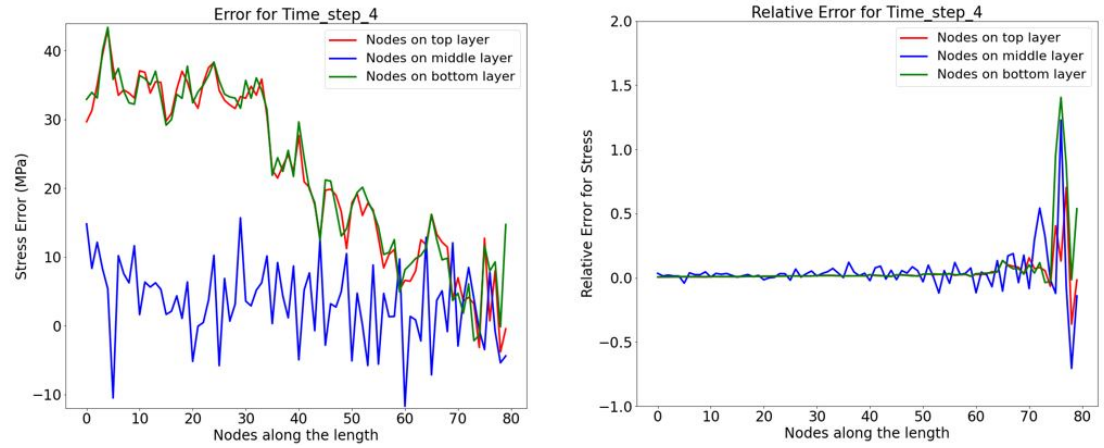

FIGURE 6.79: Error and relative error for stress at time step 4

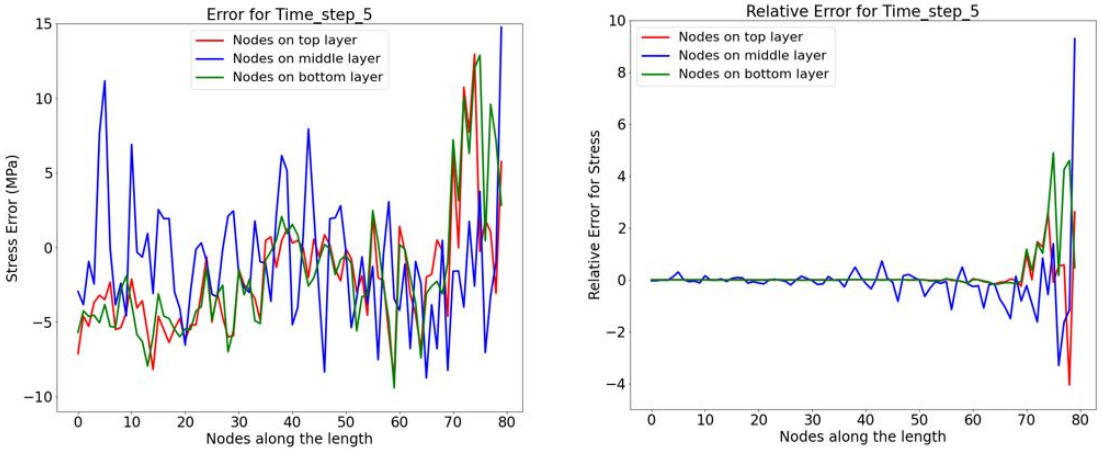

FIGURE 6.80: Error and relative error for stress at time step 5

A maximumm error of 90 MPa was observed in figure 6.76 for time step 1. All other time steps showed error below 50 MPa indicating good model performance. Also the relative error plots show close to zero value at the the critical high stress regions near the fixed end of cantilever beam.

6.2.2 Analysis with Gated Graph Architecture for Finer Mesh

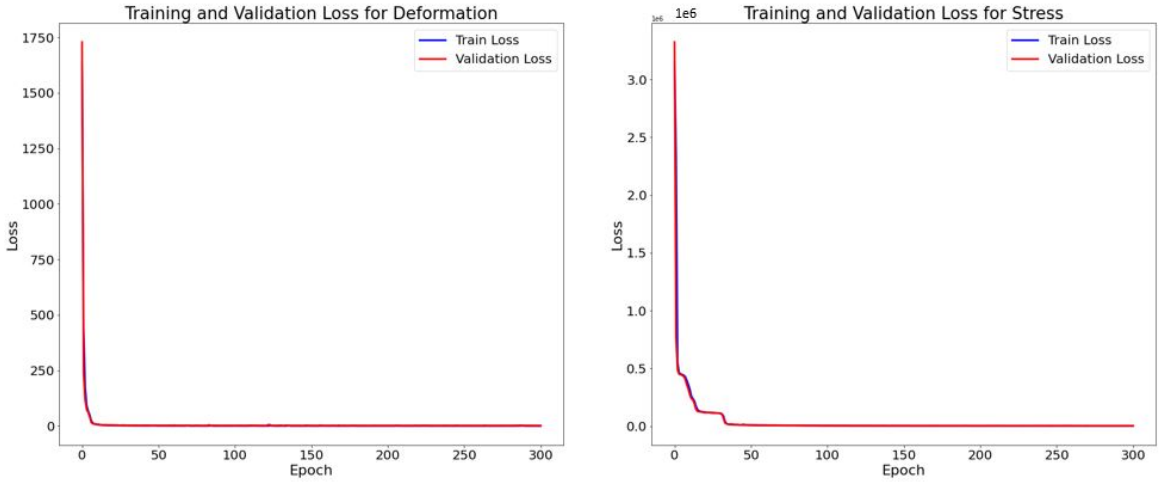

FIGURE 6.81: Training and validation curve for model with GG-NN architecture

3D Plots for Deformation

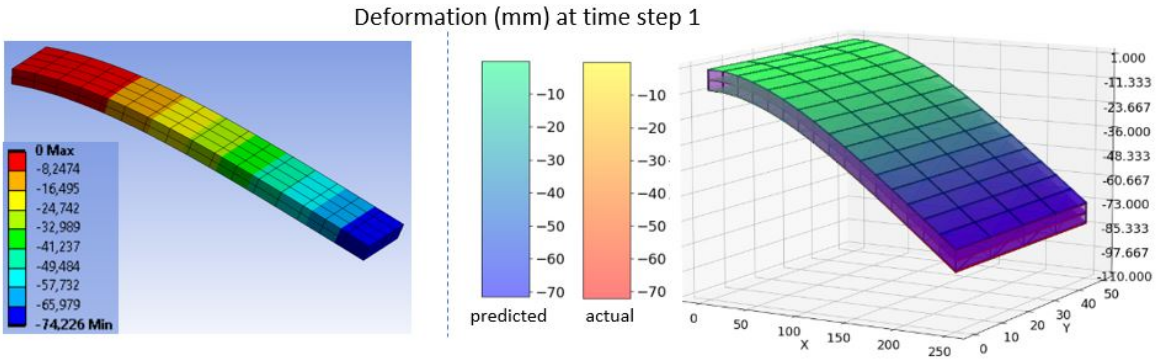

FIGURE 6.82: FEM vs. GG-NN result for deformation at time step 1

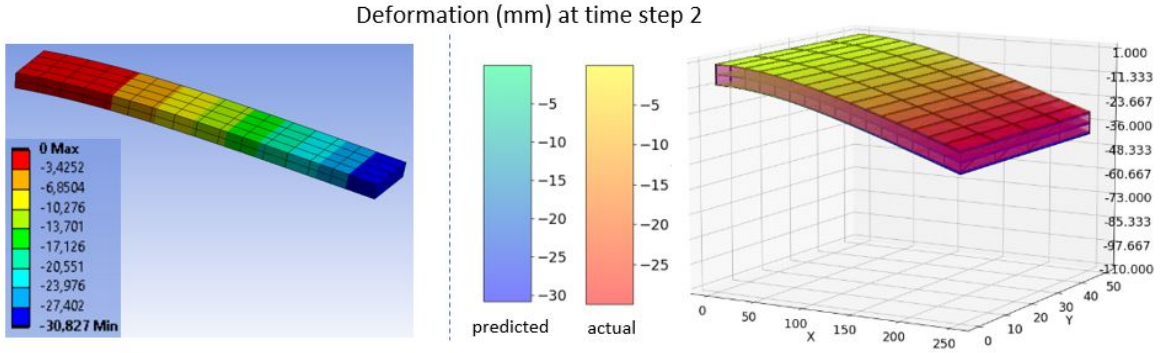

FIGURE 6.83: FEM vs. GG-NN result for deformation at time step 2

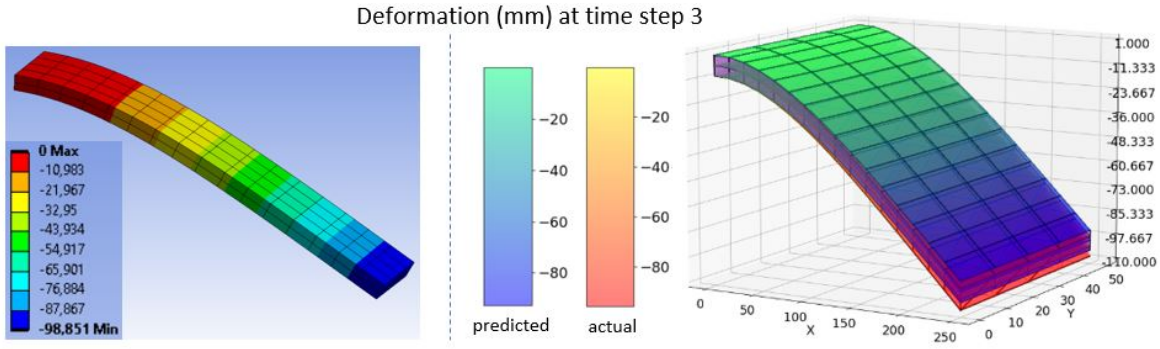

FIGURE 6.84: FEM vs. GG-NN result for deformation at time step 3

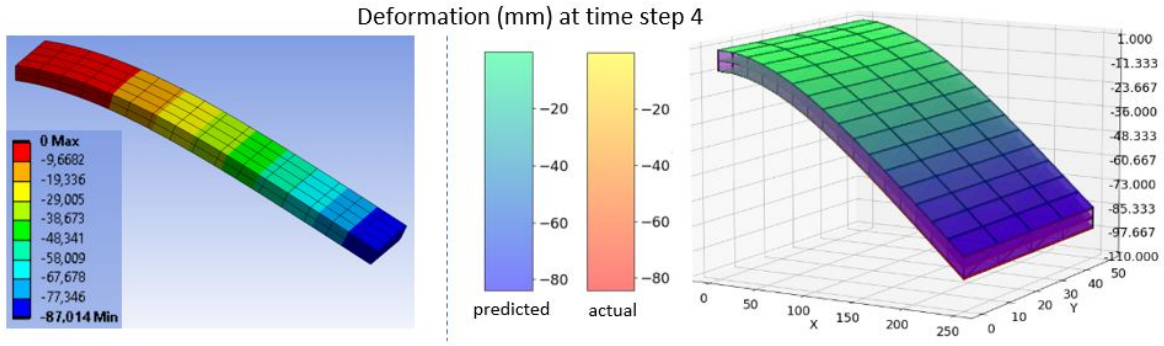

FIGURE 6.85: FEM vs. GG-NN result for deformation at time step 4

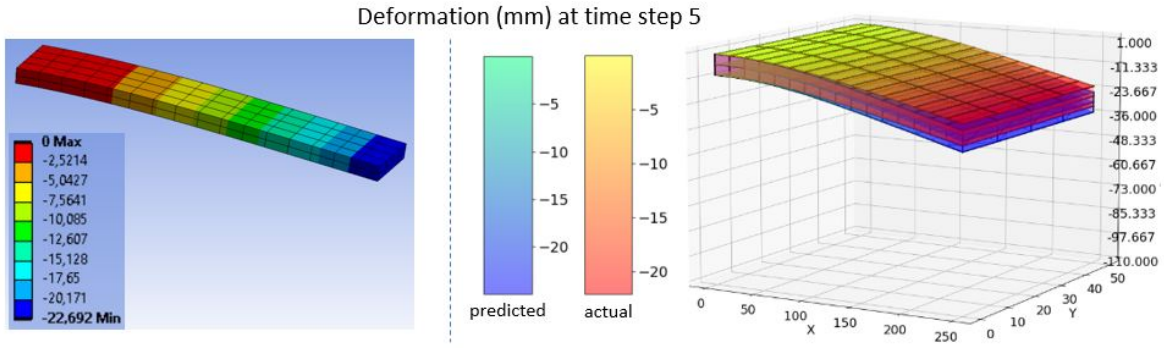

FIGURE 6.86: FEM vs. GG-NN result for deformation at time step 5

The 3D deformation plots from figure 6.82 to 6.86 show an overall good result with slight deviation on actual and predicted beams for time step 3 and 5.

The nodal deformation value plots can be found in Appendix section A.3.

Deformation Error Visualization

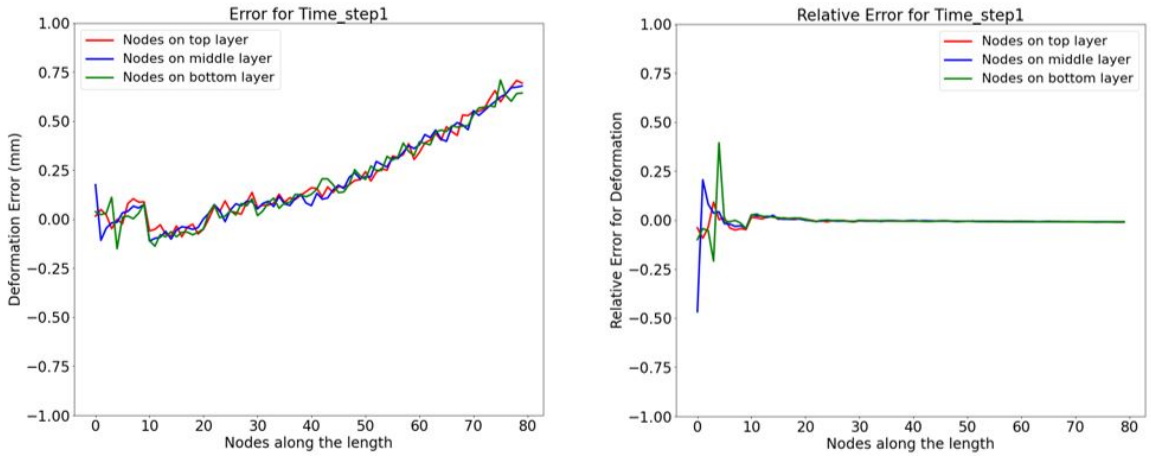

FIGURE 6.87: Error and relative error for deformation at time step 1

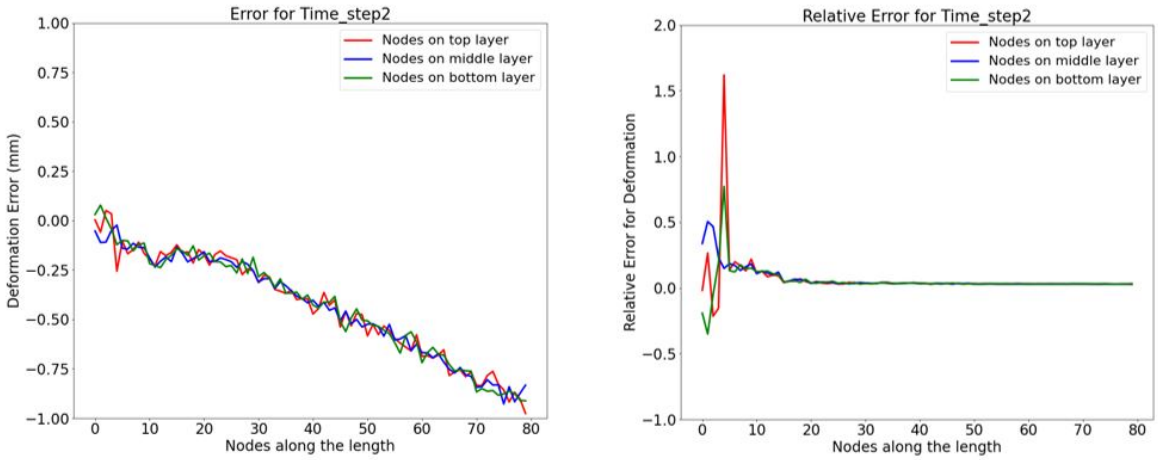

FIGURE 6.88: Error and relative error for deformation at time step 2

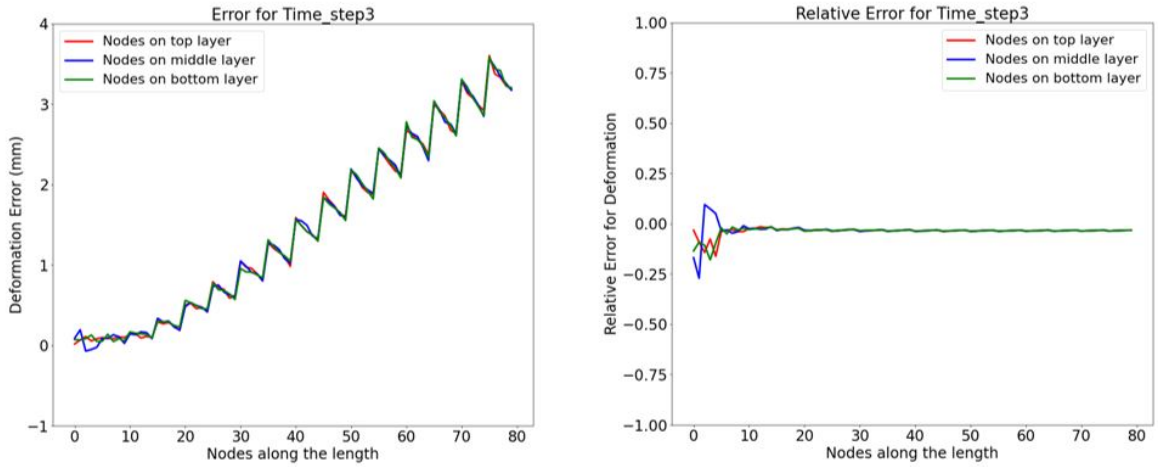

FIGURE 6.89: Error and relative error for deformation at time step 3

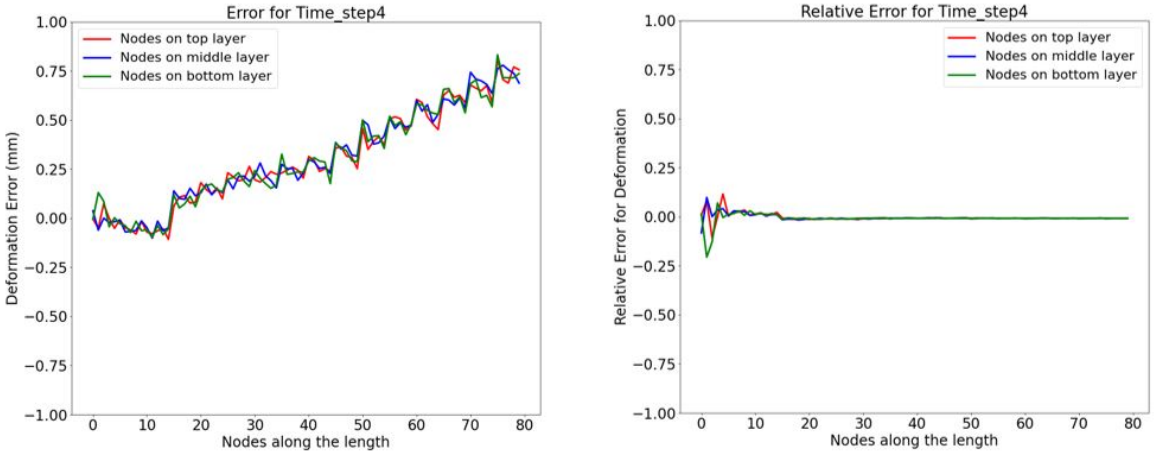

FIGURE 6.90: Error and relative error for deformation at time step 4

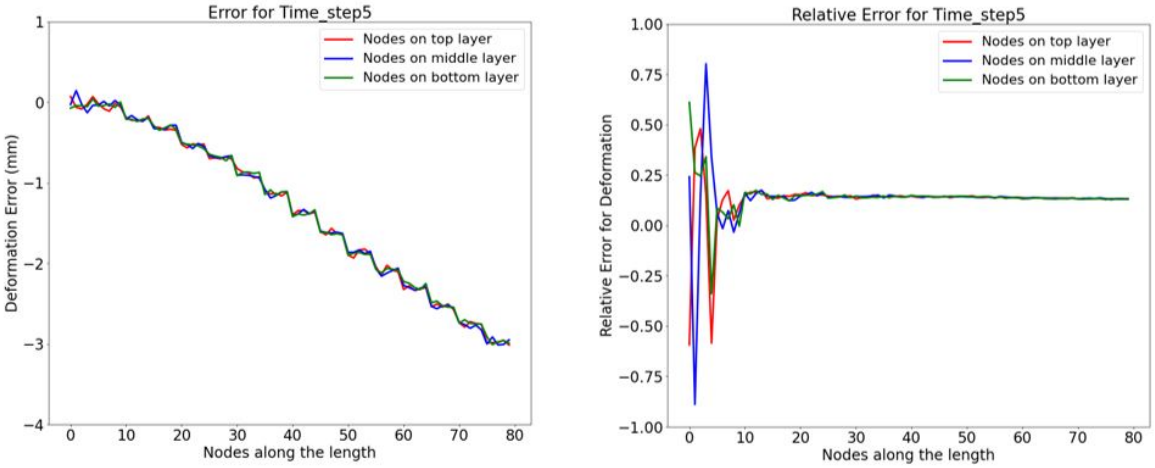

FIGURE 6.91: Error and relative error for deformation at time step 5

From the error plots in figure 6.87 to 6.91 it can be concluded that time steps 3 and 5 have max deformation error upto 4mm. For rest of the time steps the predictions are good with very less error.

3D Plots for Stress

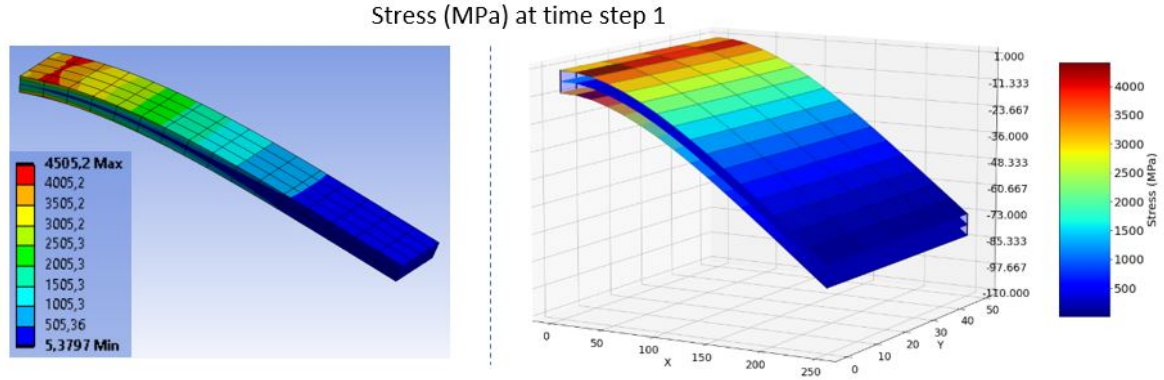

FIGURE 6.92: FEM vs. GG-NN result for stress at time step 1

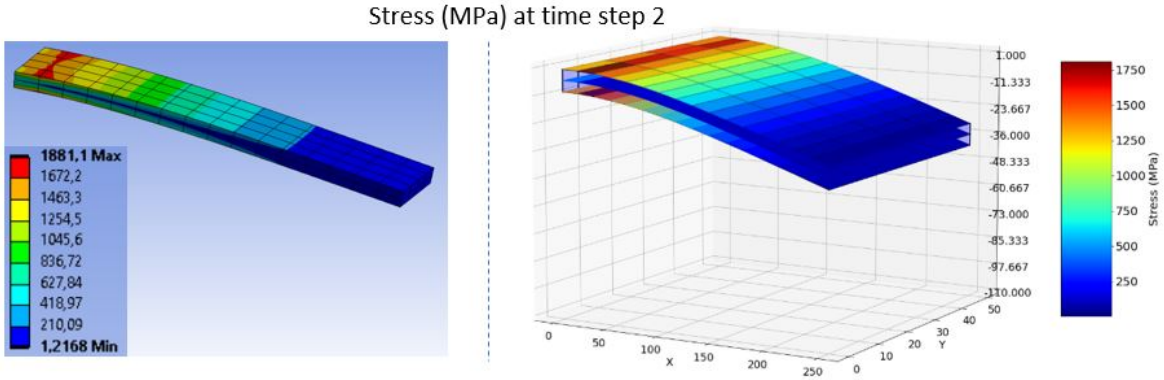

FIGURE 6.93: FEM vs. GG-NN result for stress at time step 2

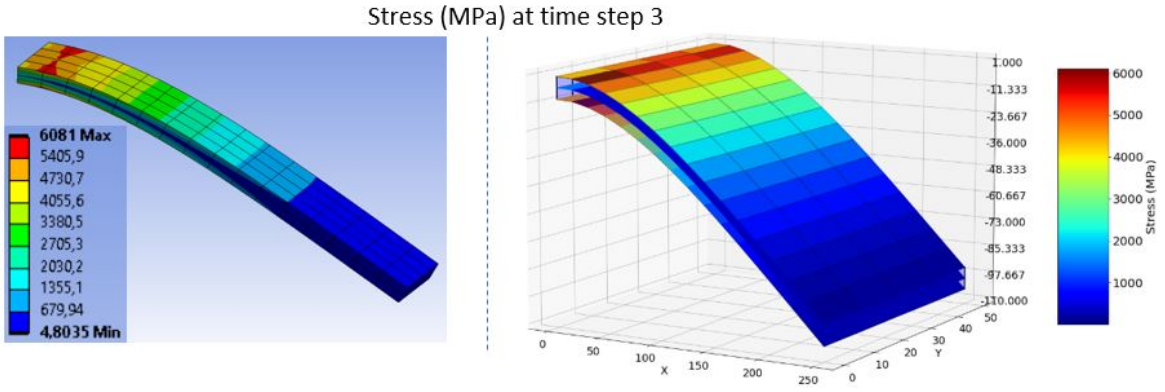

FIGURE 6.94: FEM vs. GG-NN result for stress at time step 3

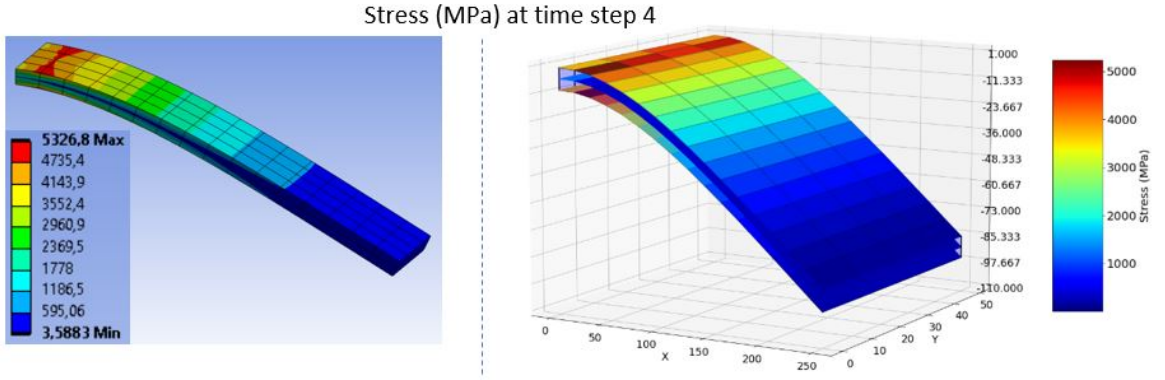

FIGURE 6.95: FEM vs. GG-NN result for stress at time step 4

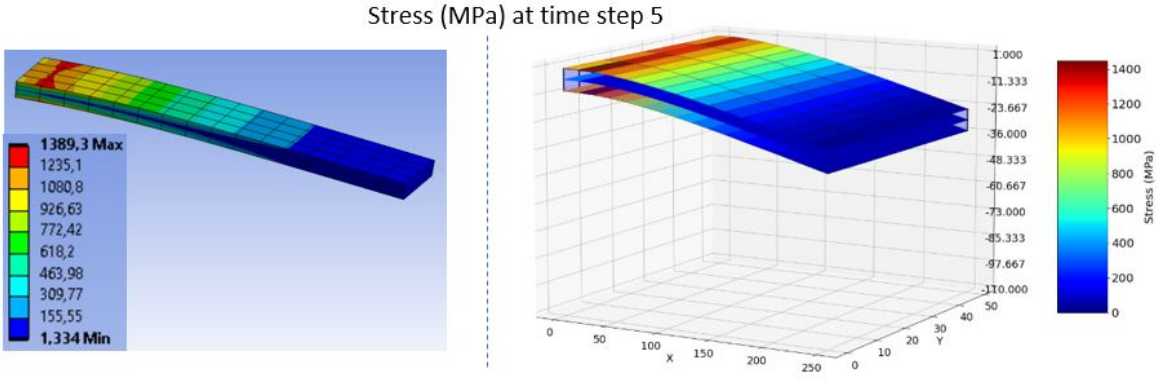

FIGURE 6.96: FEM vs. GG-NN result for stress at time step 5

Observing the plots in figure 6.92 to 6.96, the predicted stress distribution is found to be in sync with the actual values from FEA software.

The plots showing actual and predicted values for nodal stress can be seen in Appendix section A.3.

Stress Error Visualization

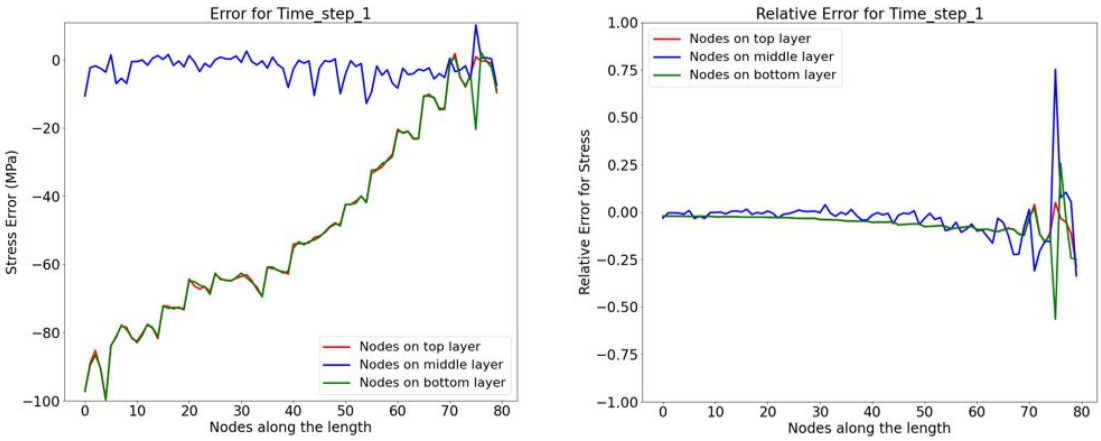

FIGURE 6.97: Error and relative error for stress at time step 1

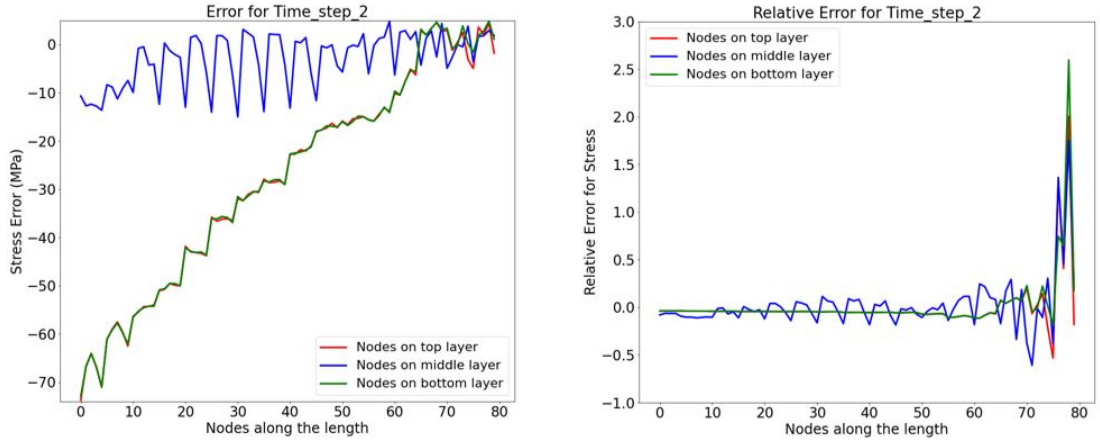

FIGURE 6.98: Error and relative error for stress at time step 2

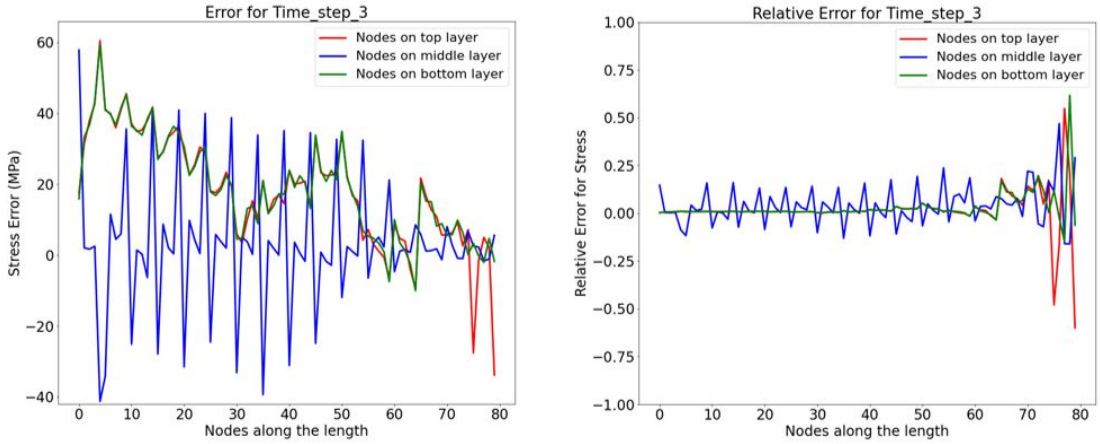

FIGURE 6.99: Error and relative error for stress at time step 3

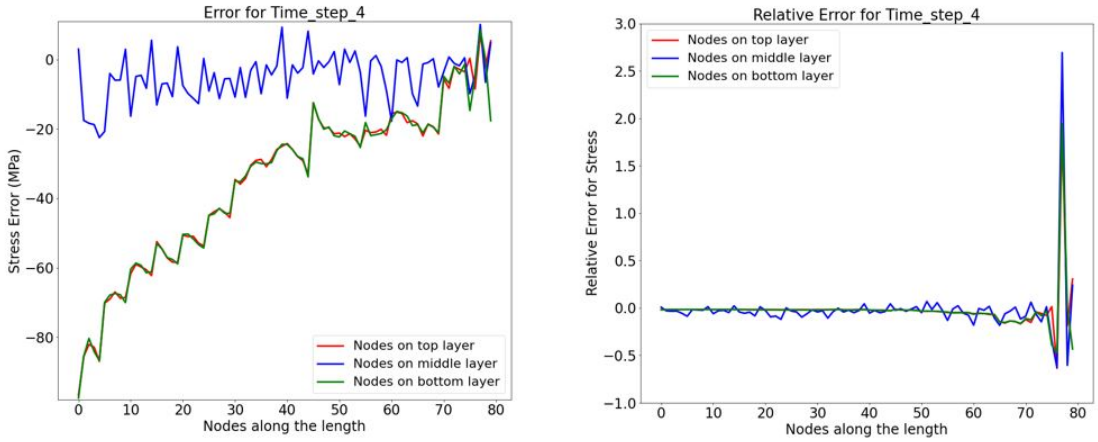

FIGURE 6.100: Error and relative error for stress at time step 4

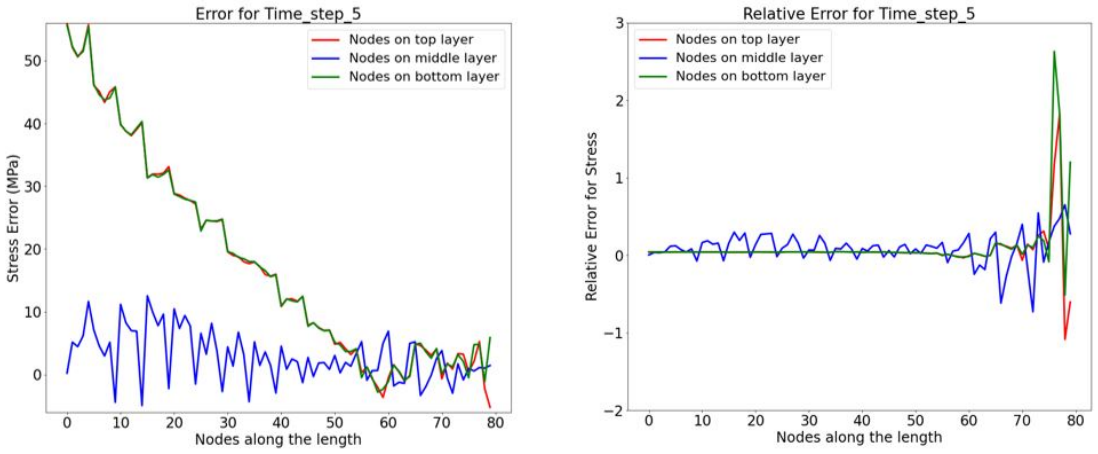

FIGURE 6.101: Error and relative error for stress at time step 5

A maximum stress error of 100 MPa can be observed for time step 1 and 4 in figure 6.97 and 6.100 respectively. Although the actual error is high , the relative error value is close to zero.

6.3 Beam with Twisting

Sometimes the boundary conditions for a practical real life example of cantilever beam would be in such a way that it might cause twisting of beam along with the bending. Therefore to generalize the cantilever beam case study it is necessary to consider forces that would cause the beam to twist in addition to the bend. The GCN and GG models were trained on twist case dataset consisting of 1000 observations. In this case, because of the twisting of beam the X and Y deformations are also predicted in addition to the Z deformation and stress values.

6.3.1 Analysis with GCN Architecture for Twist Case

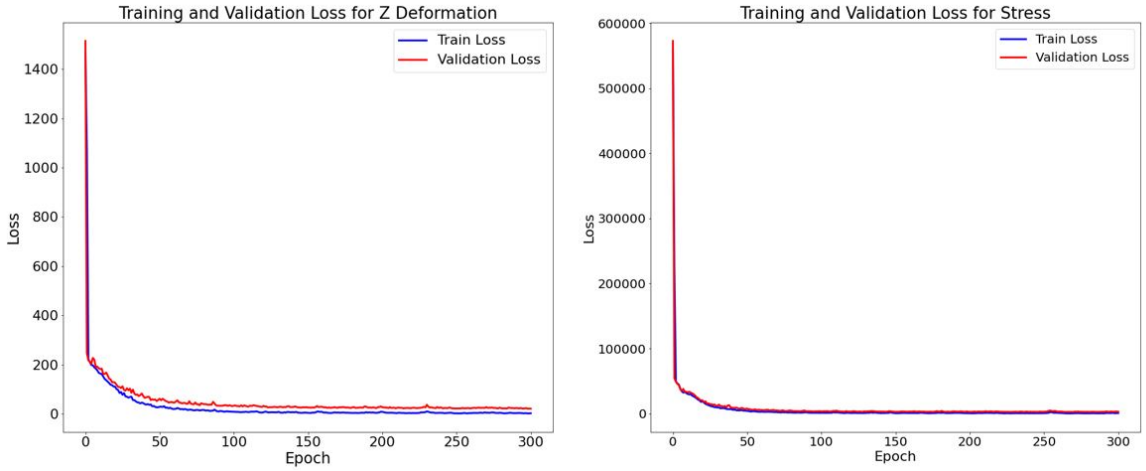

FIGURE 6.102: Training and validation curve for Z-deformation and Stress with GCN architecture

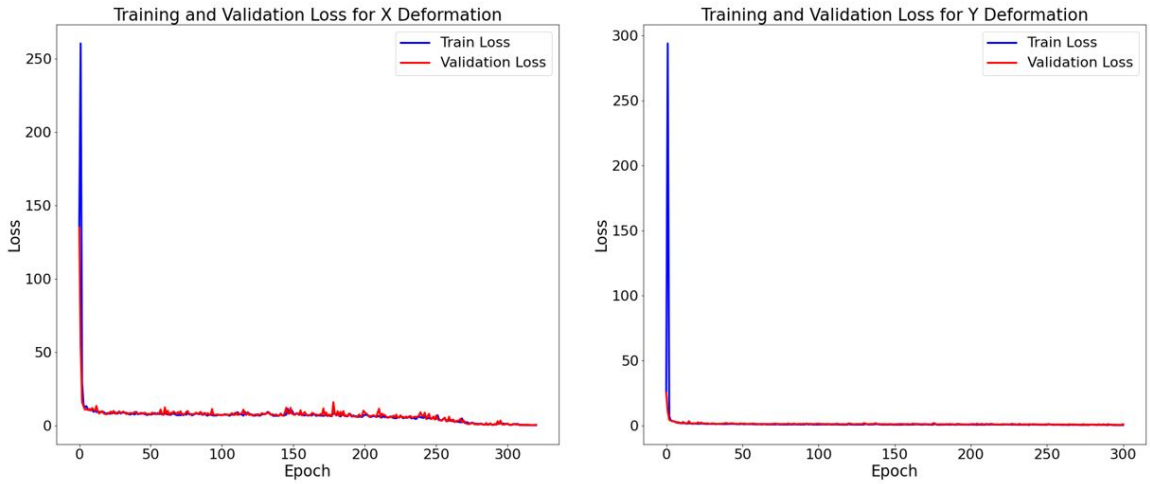

FIGURE 6.103: Training and validation curve for X and Y deformation with GCN architecture

3D Plots for Deformation

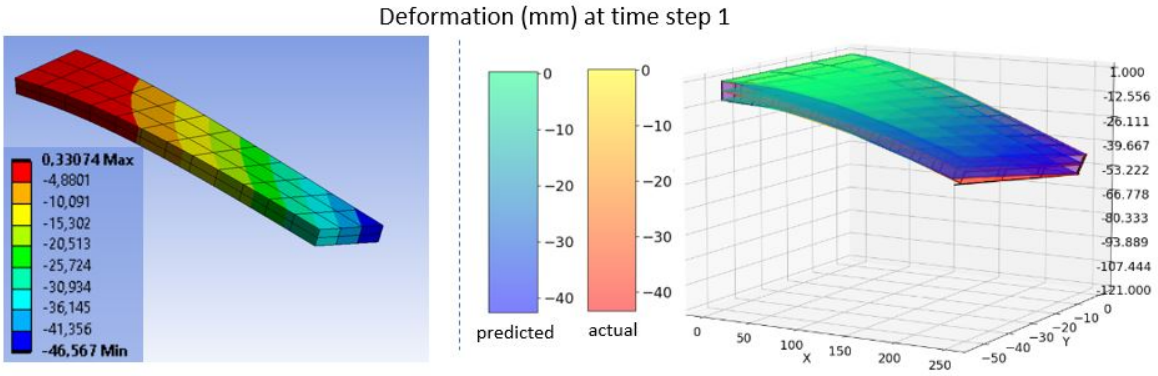

FIGURE 6.104: FEM vs. GCN result for deformation at time step 1

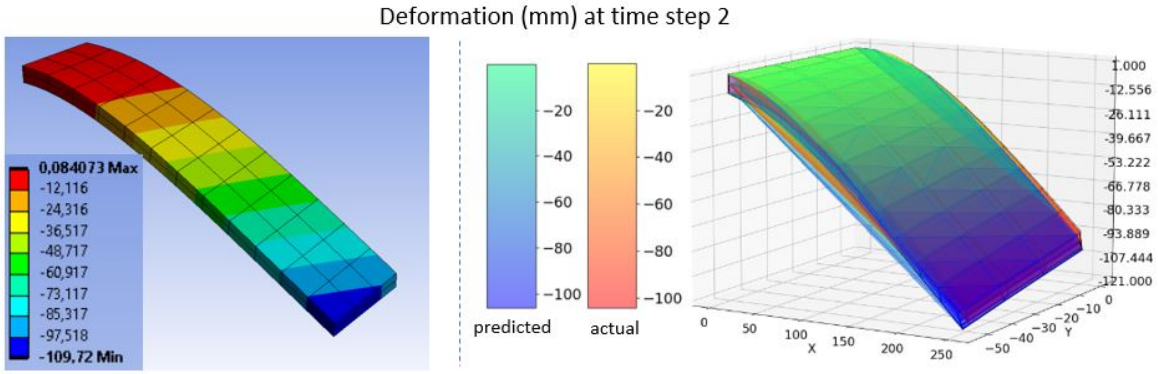

FIGURE 6.105: FEM vs. GCN result for deformation at time step 2

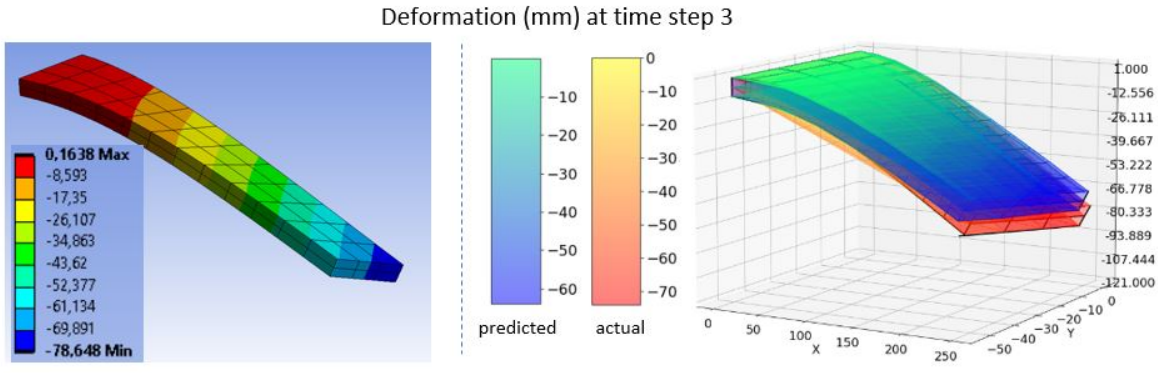

FIGURE 6.106: FEM vs. GCN result for deformation at time step 3

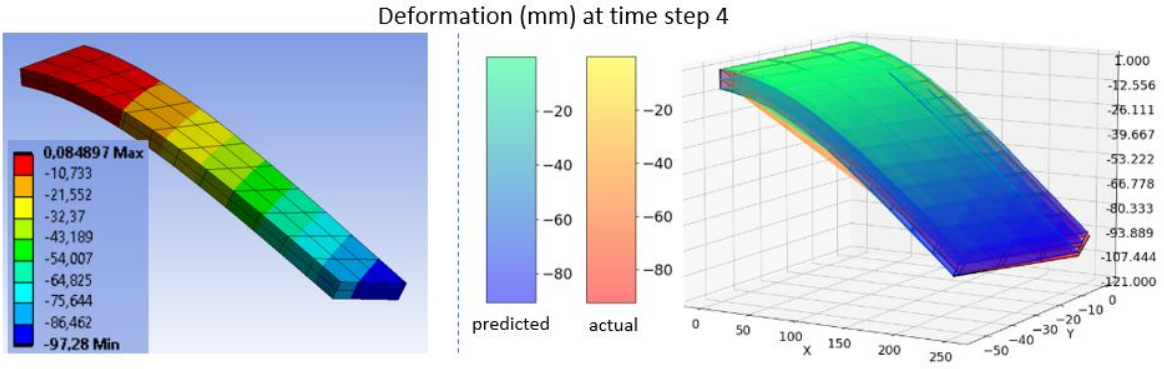

FIGURE 6.107: FEM vs. GCN result for deformation at time step 4

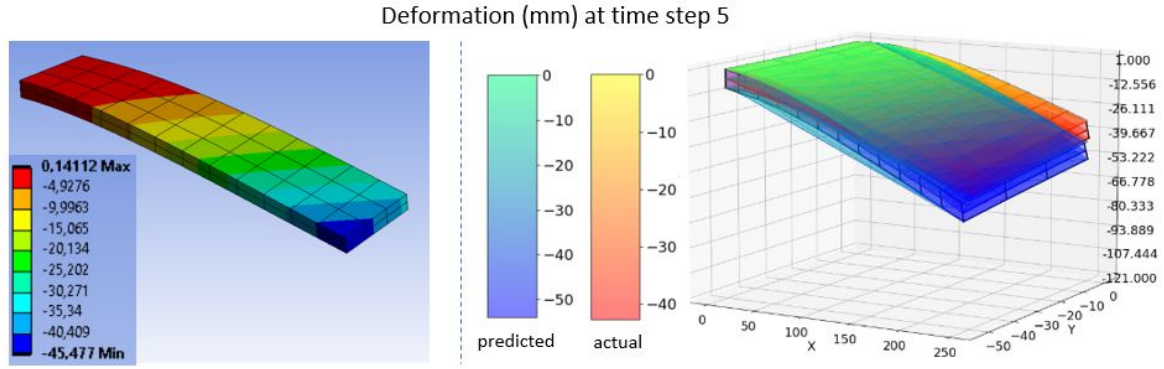

FIGURE 6.108: FEM vs. GCN result for deformation at time step 5

The 3D plots in the figure 6.104 to 6.108 show overall good GCN model performance on predicted deformation. A slight separation of actual beam and predicted beam can be observed in time steps 3 and 5.

The plots for nodal Z deformation values can be observed in Appendix A.4.

Error Visualization for Z deformation

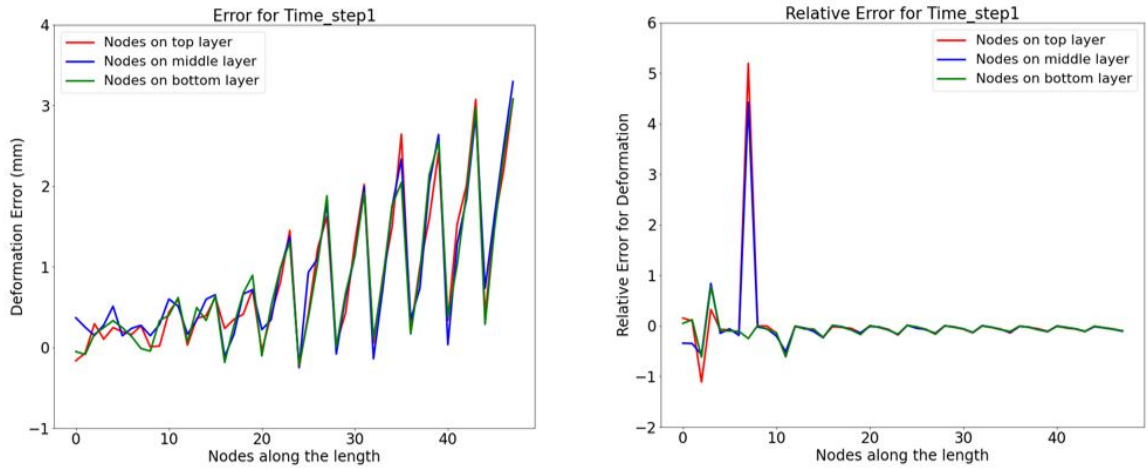

FIGURE 6.109: Error and relative error for Z deformation at time step 1

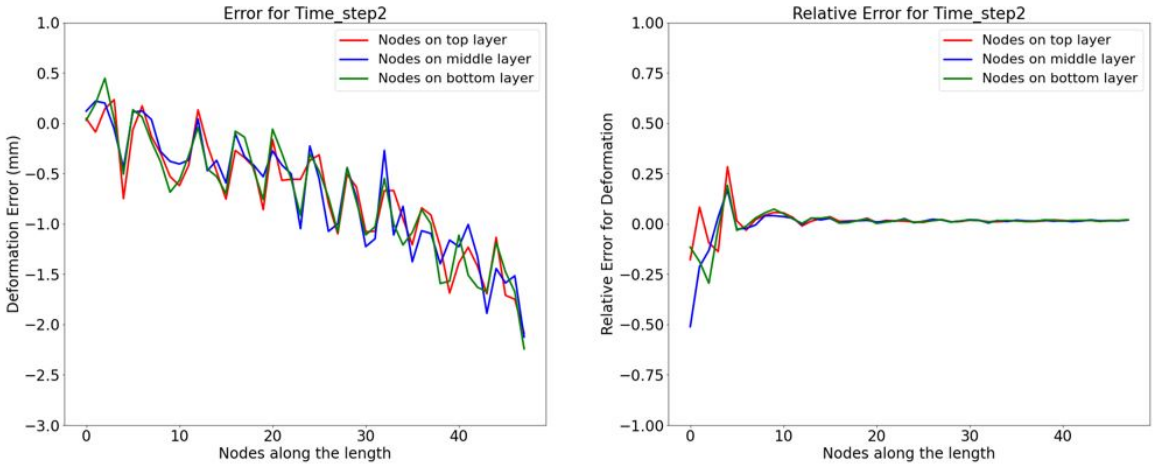

FIGURE 6.110: Error and relative error for Z deformation at time step 2

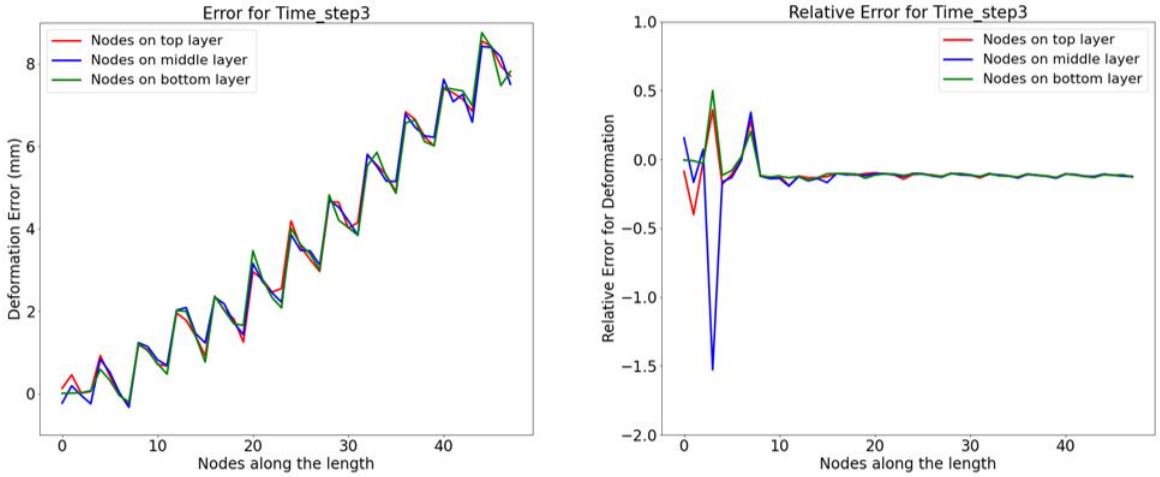

FIGURE 6.111: Error and relative error for Z deformation at time step 3

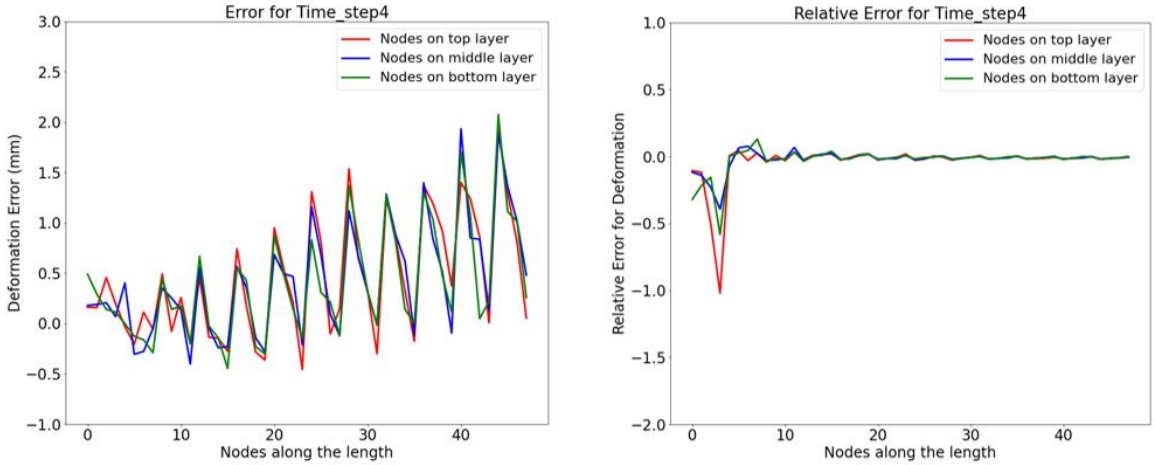

FIGURE 6.112: Error and relative error for Z deformation at time step 4

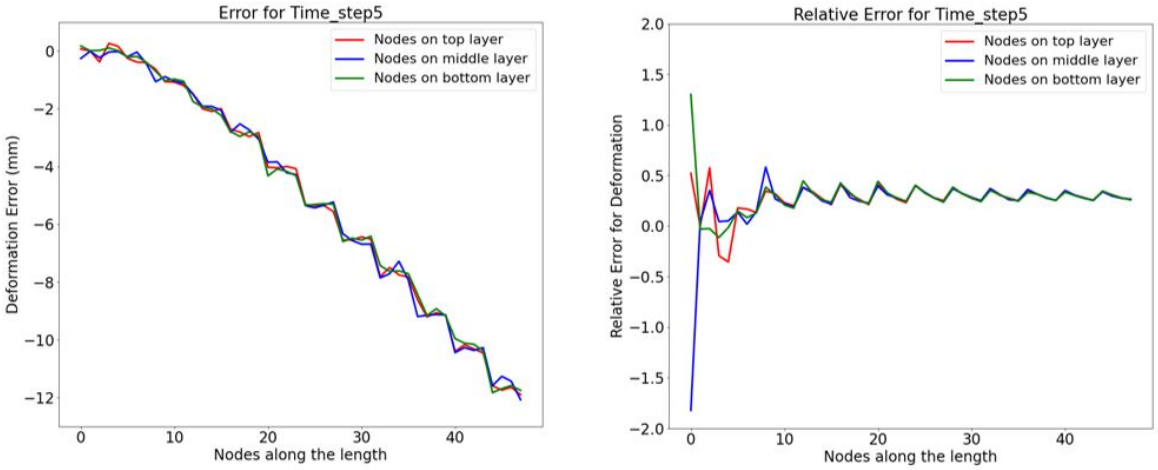

FIGURE 6.113: Error and relative error for Z deformation at time step 5

From the error plots in figure 6.111 and 6.113 it can be stated that the Z deformation error rises to 10mm and 12mm respectively at the free end of cantilever beam confirming to the offset observed in the 3D deformation plots. The shift in relative error line away from zero in time steps 3 and 5 indicate the significance of the offset.

Error Visualization for X and Y deformation

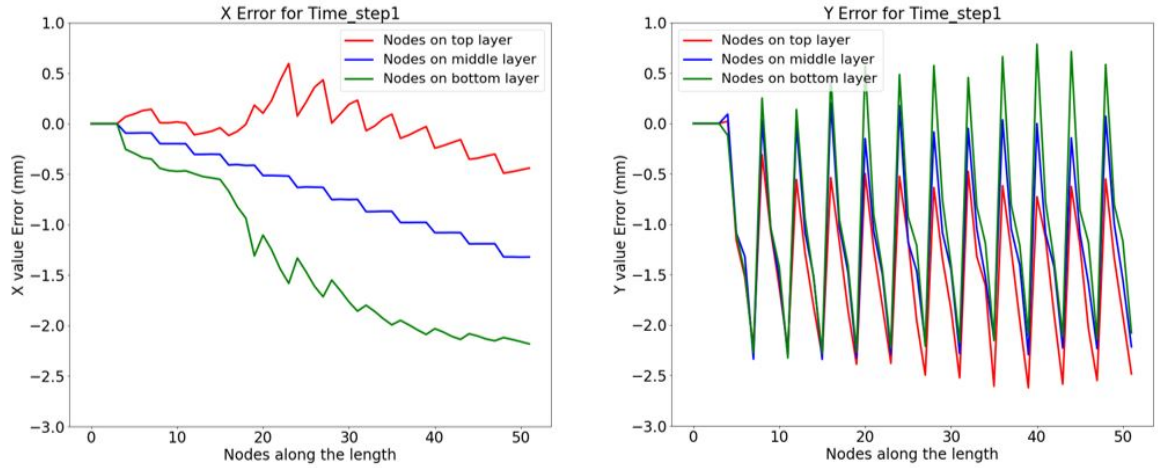

FIGURE 6.114: Error for X and Y deformation at time step 1

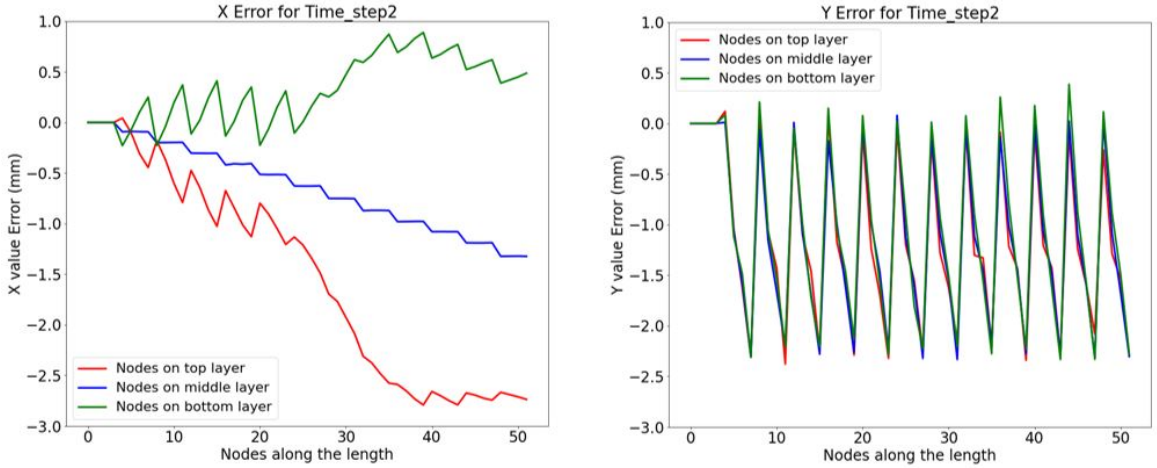

FIGURE 6.115: Error for X and Y deformation at time step 2

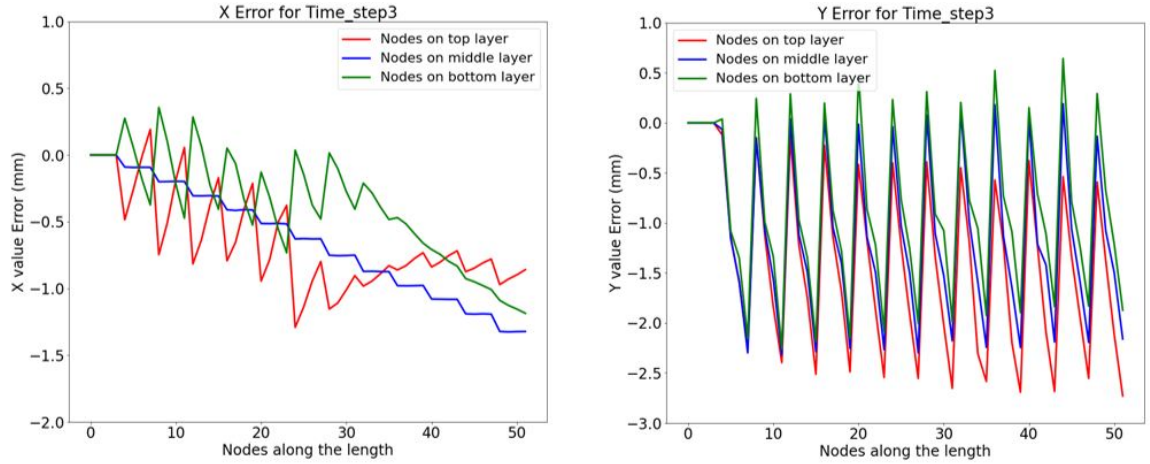

FIGURE 6.116: Error for X and Y deformation at time step 3

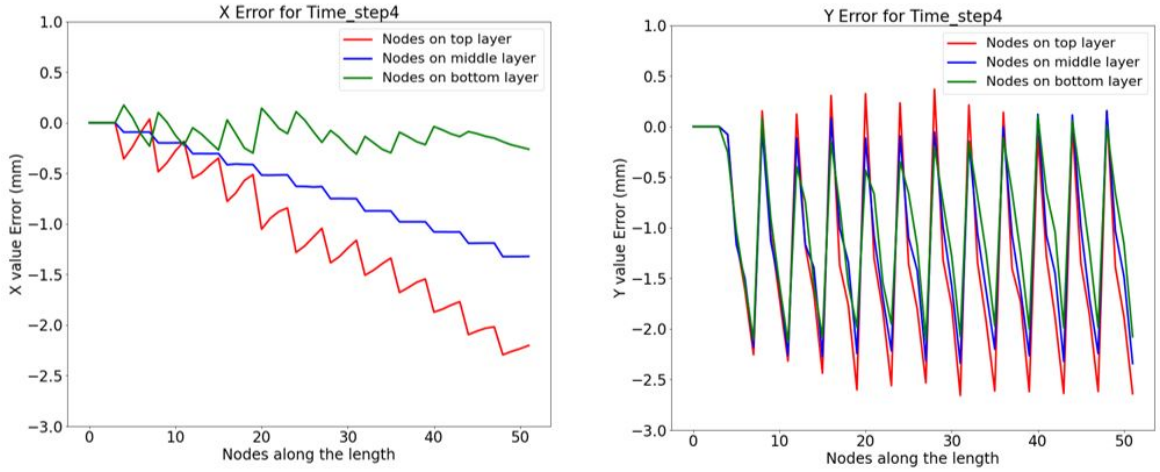

FIGURE 6.117: Error for X and Y deformation at time step 4

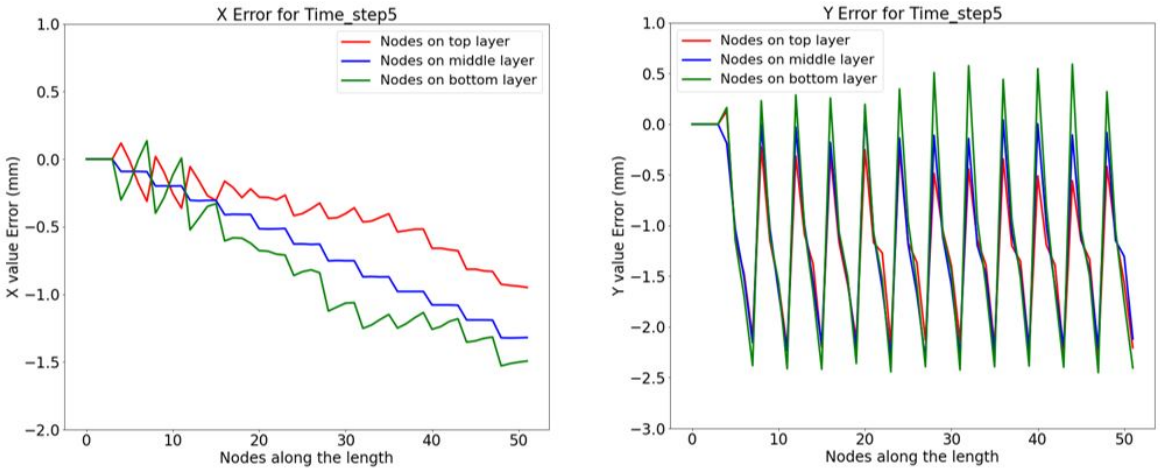

FIGURE 6.118: Error for X and Y deformation at time step 5

Figure 6.114 to 6.118 indicate the deviation in X and Y value prediction for the nodal points. The X axis runs parallel to the length while the Y axis runs parallel to the lateral width of the beam. The maximum deviation around 2.5mm for X and Y values can be seen over the time steps from the plots.

3D Plots for Stress

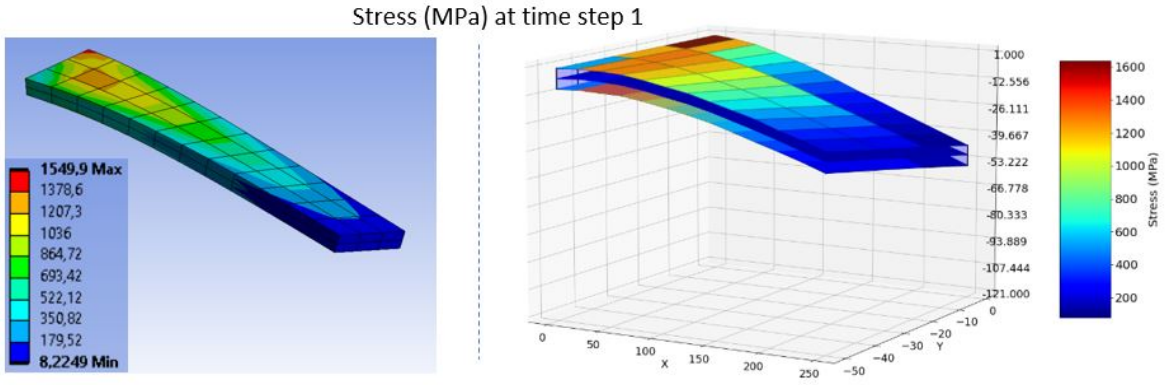

FIGURE 6.119: FEM vs. GCN result for stress at time step 1

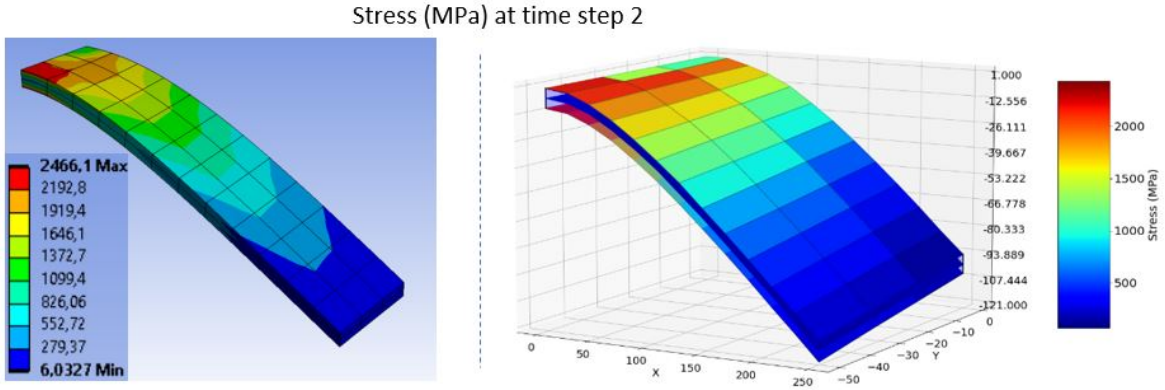

FIGURE 6.120: FEM vs. GCN result for stress at time step 2

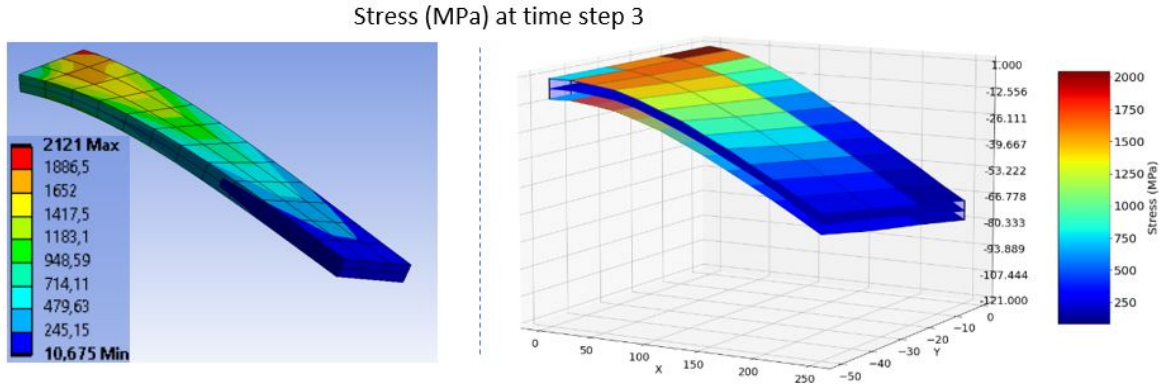

FIGURE 6.121: FEM vs. GCN result for stress at time step 3

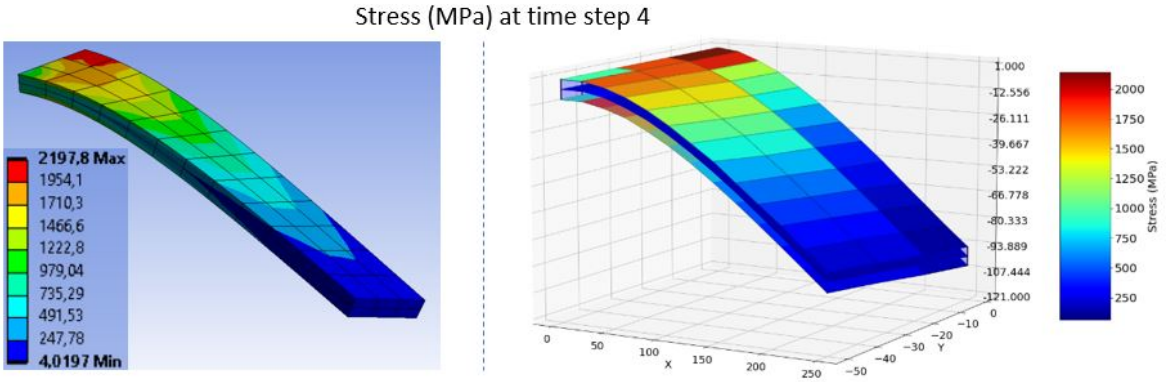

FIGURE 6.122: FEM vs. GCN result for stress at time step 4

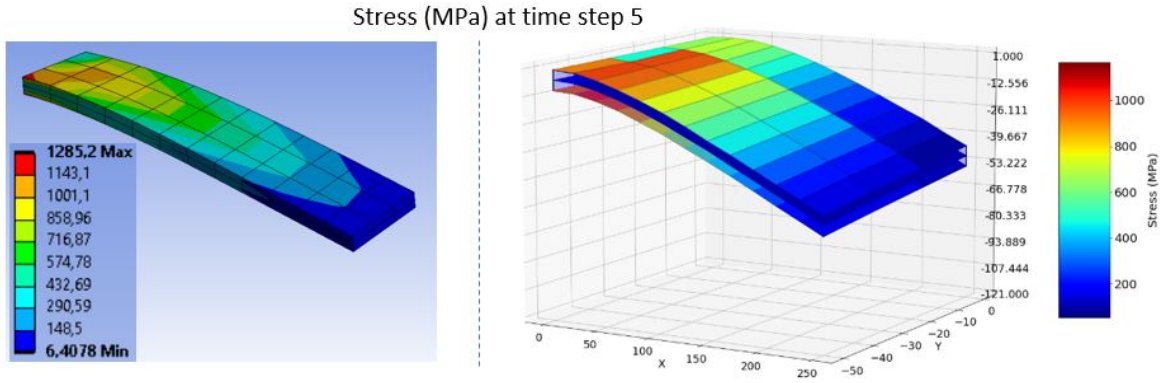

FIGURE 6.123: FEM vs. GCN result for stress at time step 5

The twisting of beam towards left or right changes the region of maximum induced stress on the beam. This can be rightly seen in the FEA results on the left from figure 6.119 to 6.123. The corresponding 3D plots on the right which indicate the results obtained from the model show the exact change in nature similar to the results from the FEA software. The GCN model is thus proficient in predicting the stress values at different locations on the beam.

The plots for actual nodal stress values can be seen in Appendix section A.4.

Stress Error Visualization

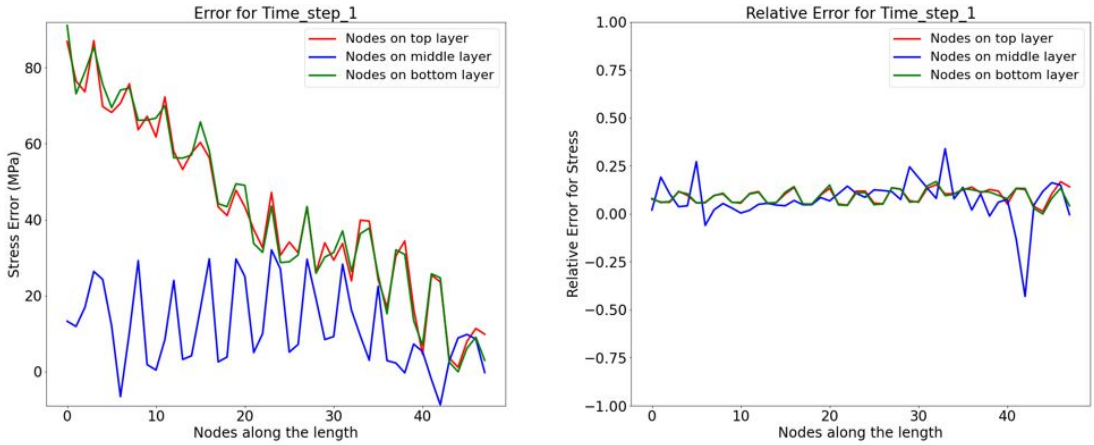

FIGURE 6.124: Error and relative error for stress at time step 1

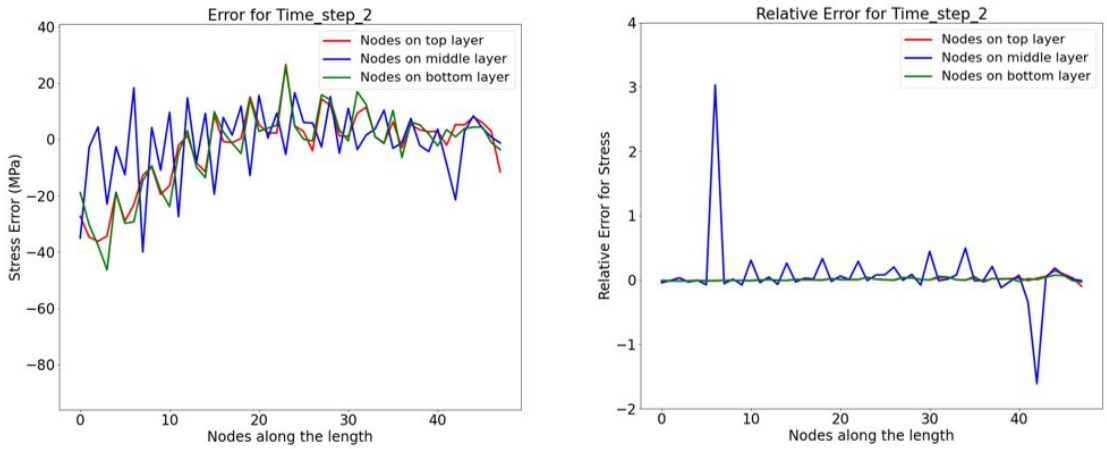

FIGURE 6.125: Error and relative error for stress at time step 2

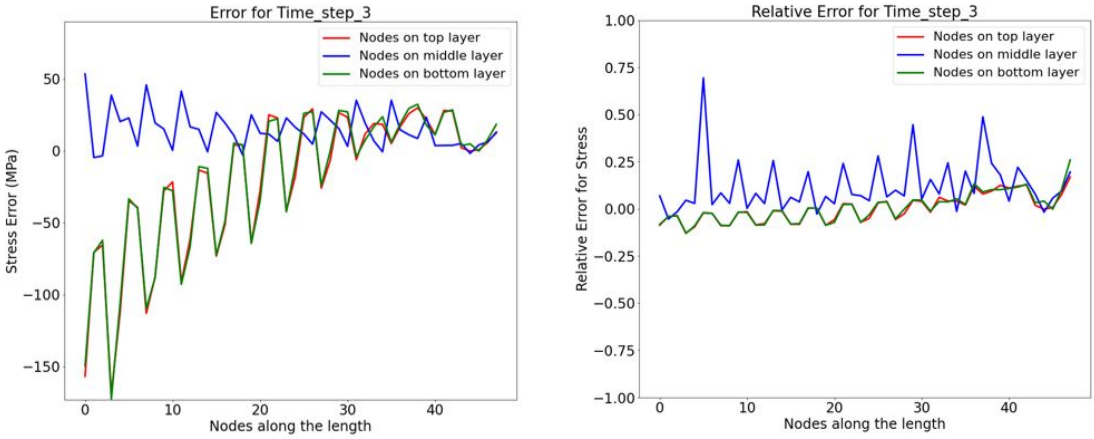

FIGURE 6.126: Error and relative error for stress at time step 3

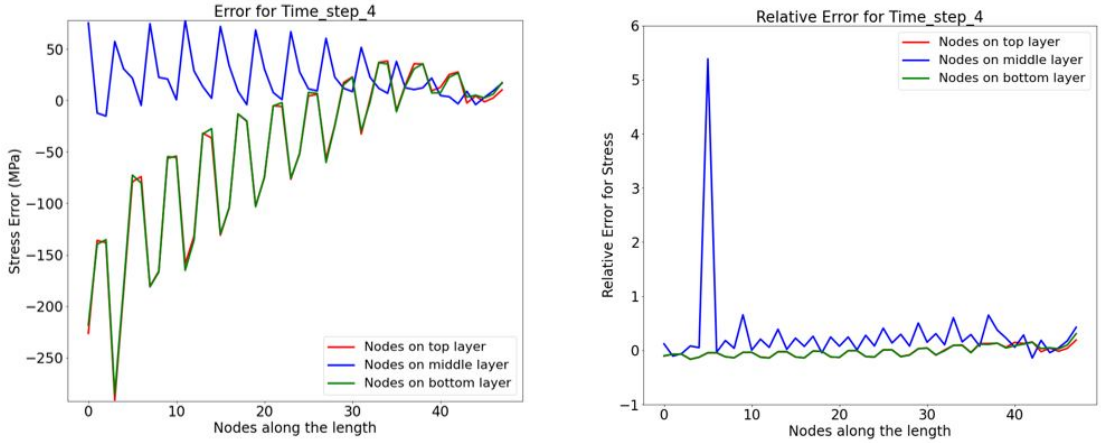

FIGURE 6.127: Error and relative error for stress at time step 4

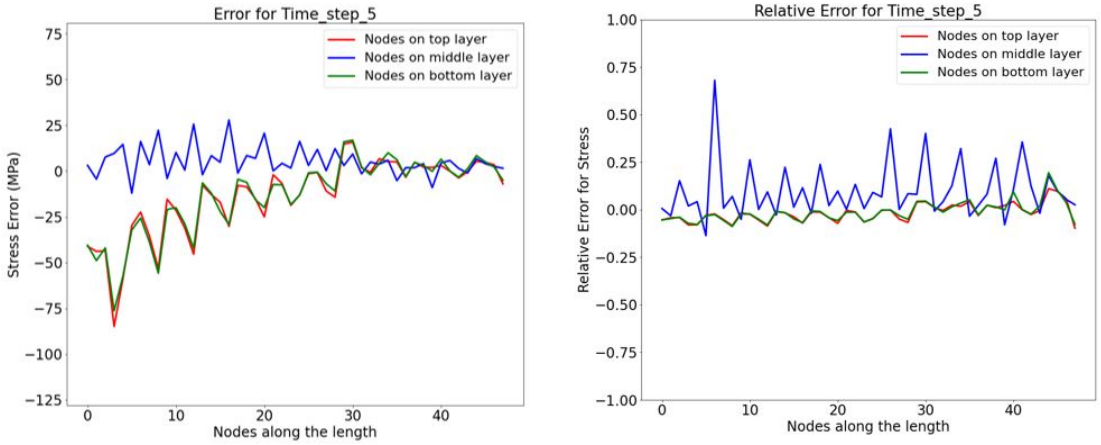

FIGURE 6.128: Error and relative error for stress at time step 5

The nodal stress error plots from figure 6.124 to 6.128 indicate that apart from time step 4 where high error of around 250 MPa is observed, the other time steps show comparatively moderate error range implying satisfactory results.

6.3.2 Analysis with Gated Graph Architecture for Twist Case

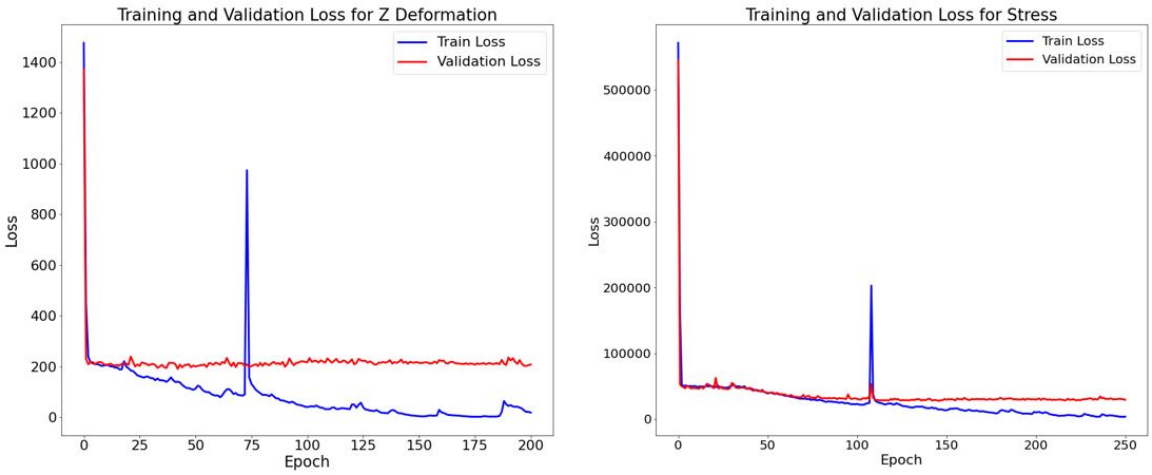

FIGURE 6.129: Training and validation curve for Z-deformation and Stress with GG-NN architecture

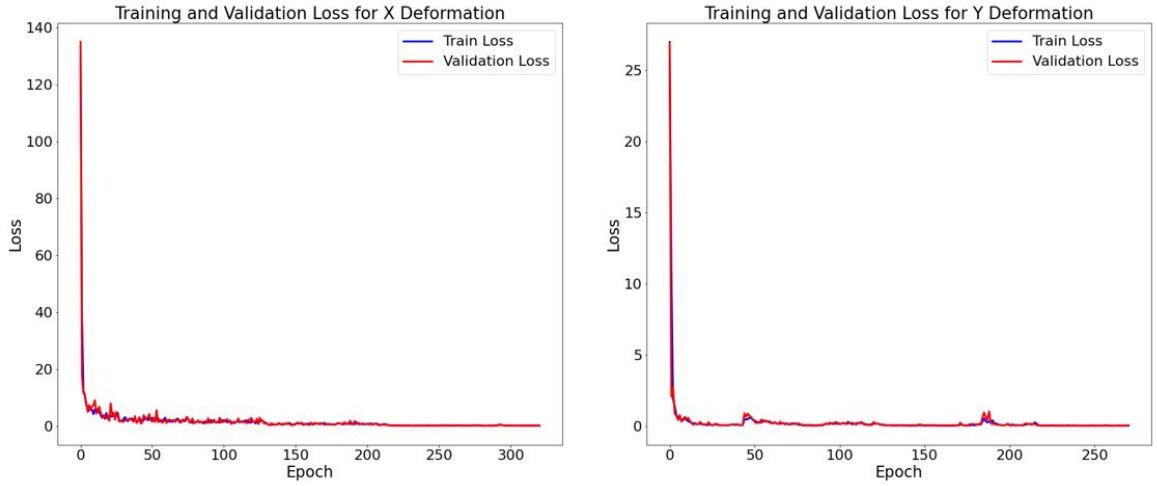

FIGURE 6.130: Training and validation curve for X and Y deformation with GG-NN architecture

The learning curve for Z deformation and stress indicate that the gated graph model fails to converge and generalize the relation between input forces and output Z deformation and stress. Also these results were after rigorous tuning of hyperparameters and creating a robust model which implies that the GG-NN architecture is not very efficient when compared to GCN architecture for this twist case. This is reflected from the following results.

3D Plots for Deformation

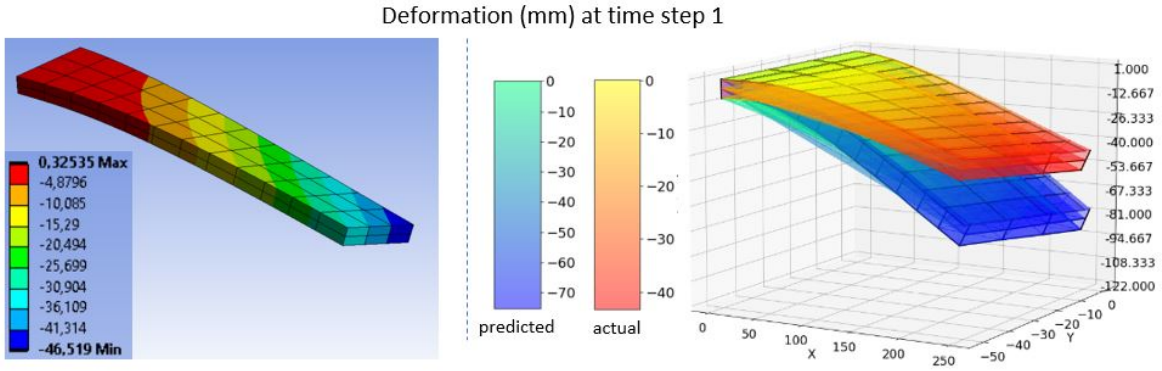

FIGURE 6.131: FEM vs. GG-NN result for deformation at time step 1

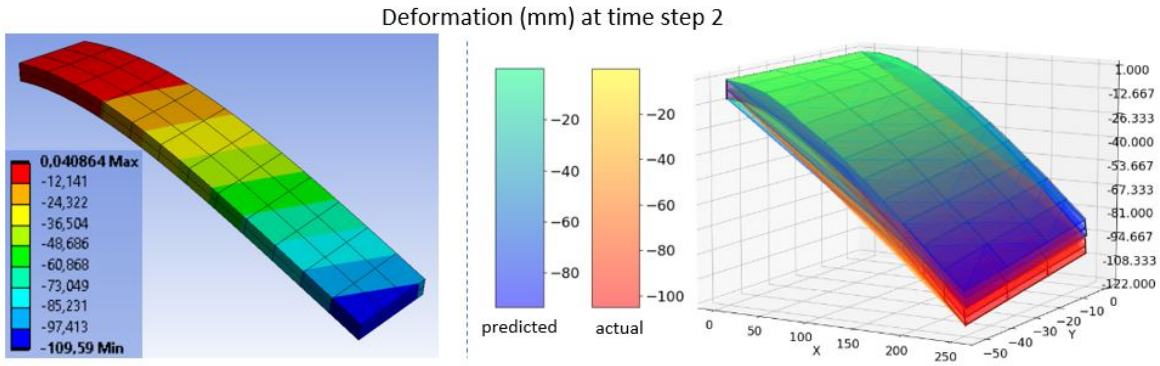

FIGURE 6.132: FEM vs. GG-NN result for deformation at time step 2

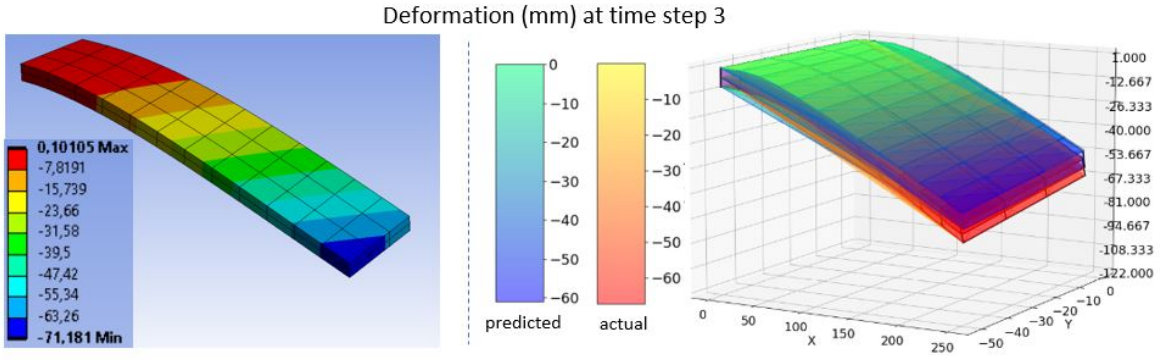

FIGURE 6.133: FEM vs. GG-NN result for deformation at time step 3

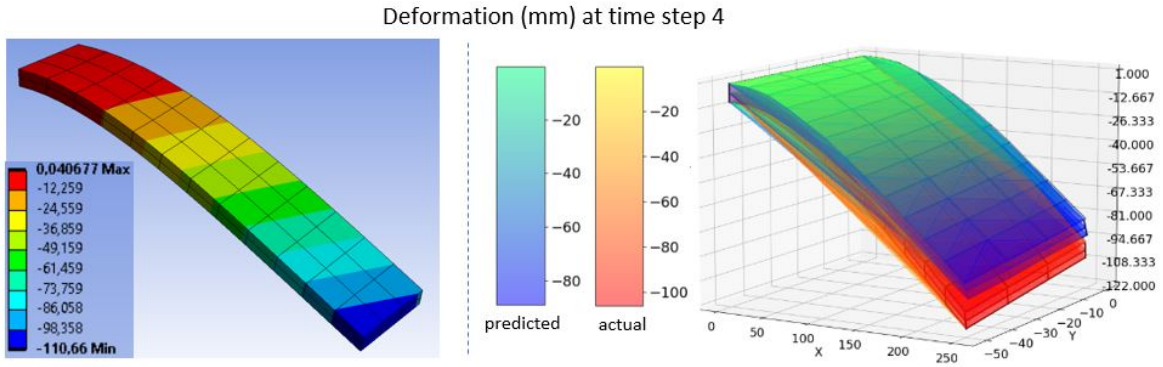

FIGURE 6.134: FEM vs. GG-NN result for deformation at time step 4

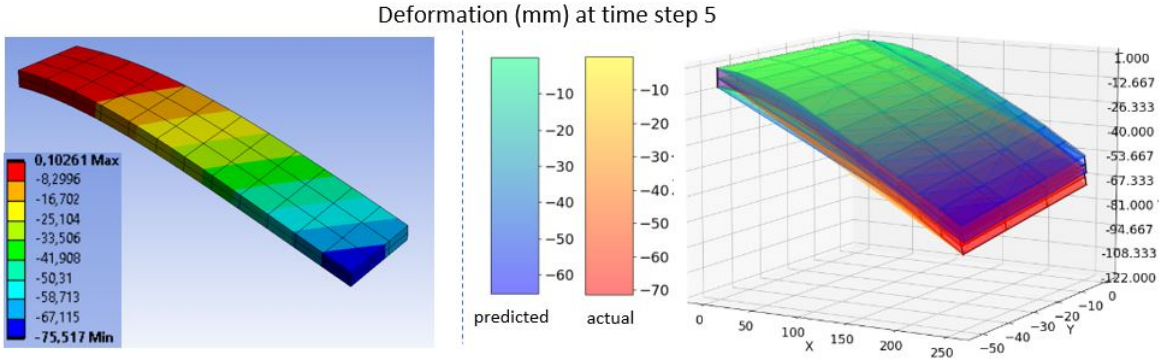

FIGURE 6.135: FEM vs. GG-NN result for deformation at time step 5

From figure 6.131 to 6.135, looking at he high offset between actual and predicted beam, it is clear that the GG-NN model fails to generalize the predictions for Z deformation. Almost in every time step there is notable separation between the actual and predicted beam indicating below par performance and less suitability of GG-NN model for the twist case.

The plots for nodal Z deformation value can be seen in Appendix section A.5.

Error Visualization for Z deformation

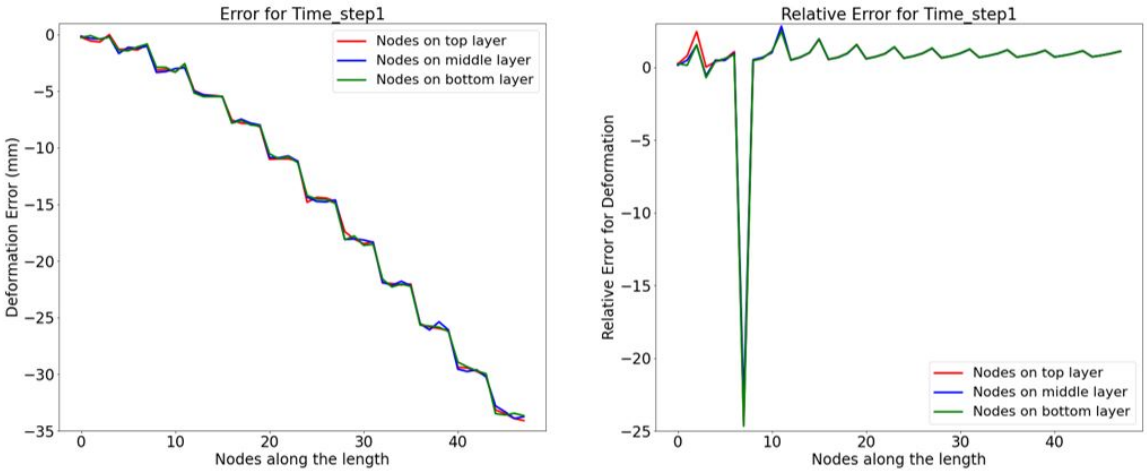

FIGURE 6.136: Error and relative error for Z deformation at time step 1

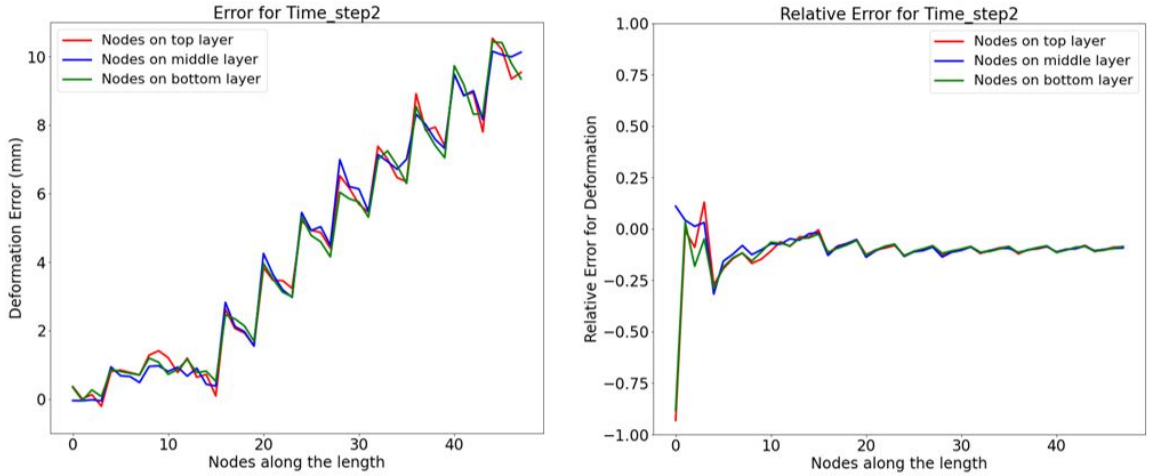

FIGURE 6.137: Error and relative error for Z deformation at time step 2

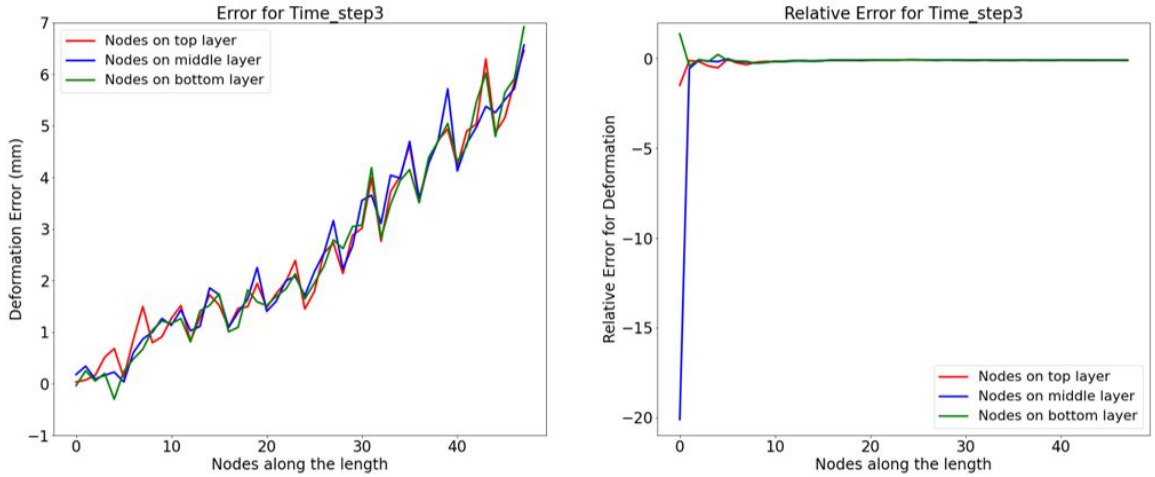

FIGURE 6.138: Error and relative error for Z deformation at time step 3

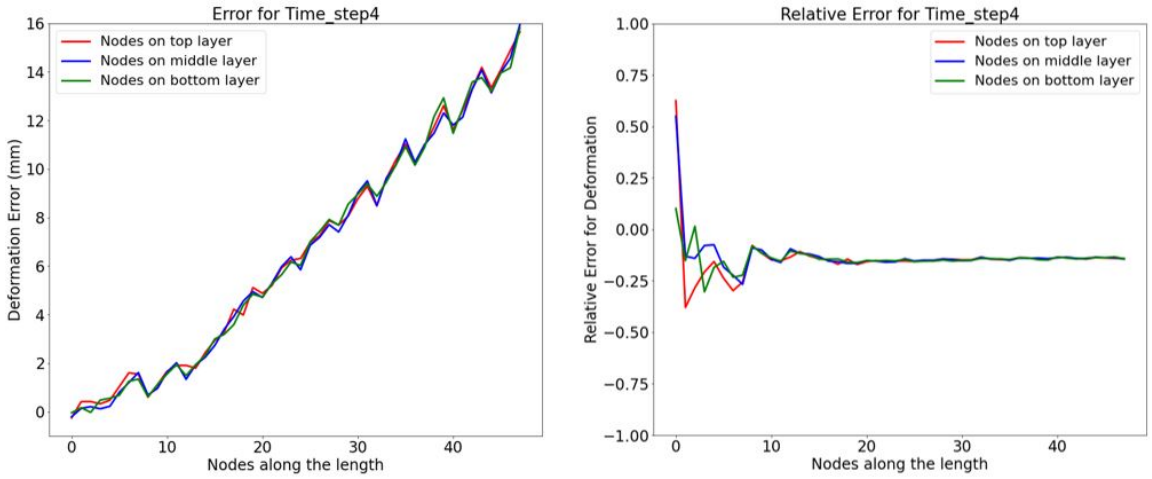

FIGURE 6.139: Error and relative error for Z deformation at time step 4

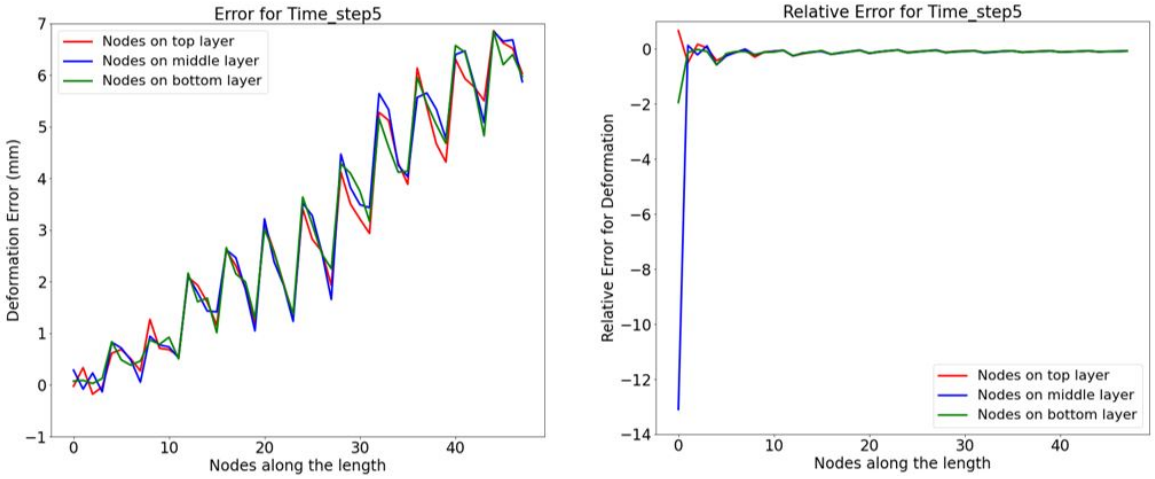

FIGURE 6.140: Error and relative error for Z deformation at time step 5

The high Z deformation error observed in figure 6.136 and 6.139 confirm to the wide separation of beams from the 3D plots. Also observing the errors in other time step plots it can be generalized that GG-NN model gives below par performance.

The Error plots for nodal X and Y value predictions can be seen in Appendix section A.5.

3D Plots for Stress

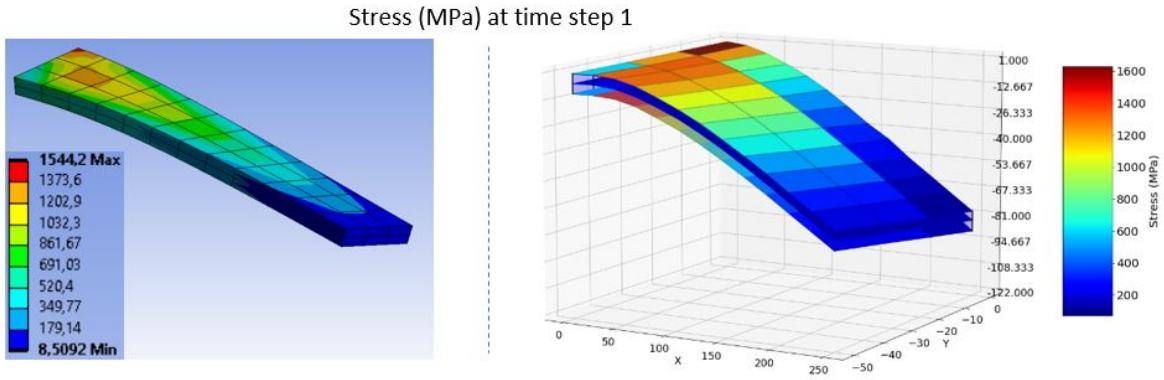

FIGURE 6.141: FEM vs. GG-NN result for stress at time step 1

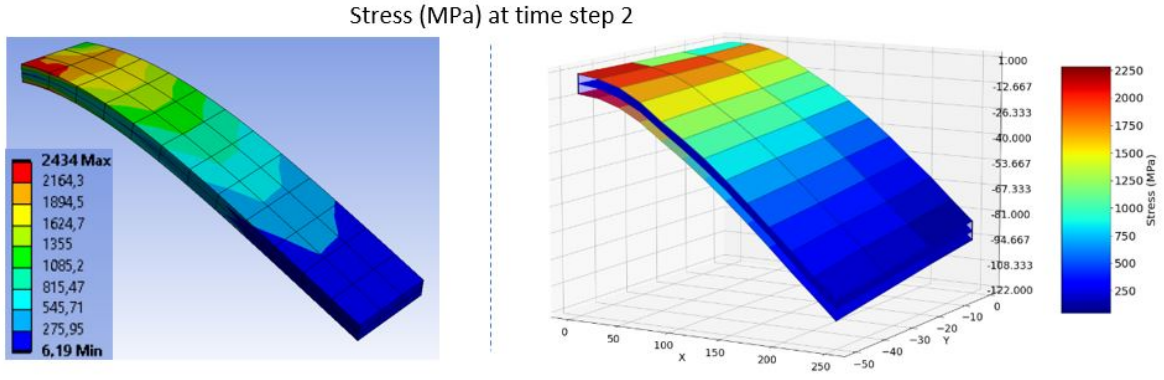

FIGURE 6.142: FEM vs. GG-NN result for stress at time step 2

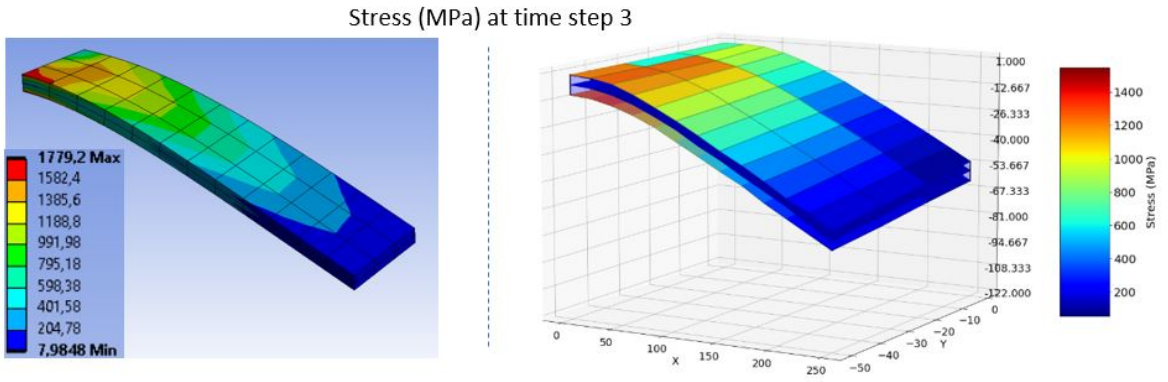

FIGURE 6.143: FEM vs. GG-NN result for stress at time step 3

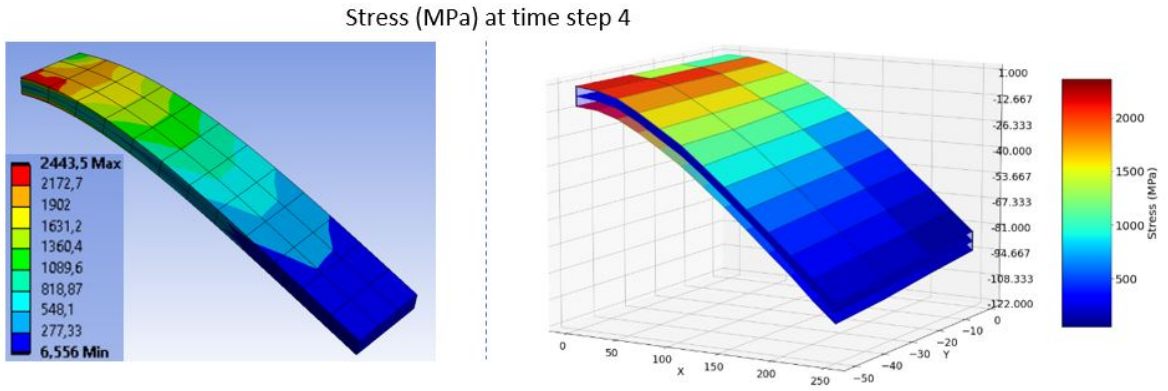

FIGURE 6.144: FEM vs. GG-NN result for stress at time step 4

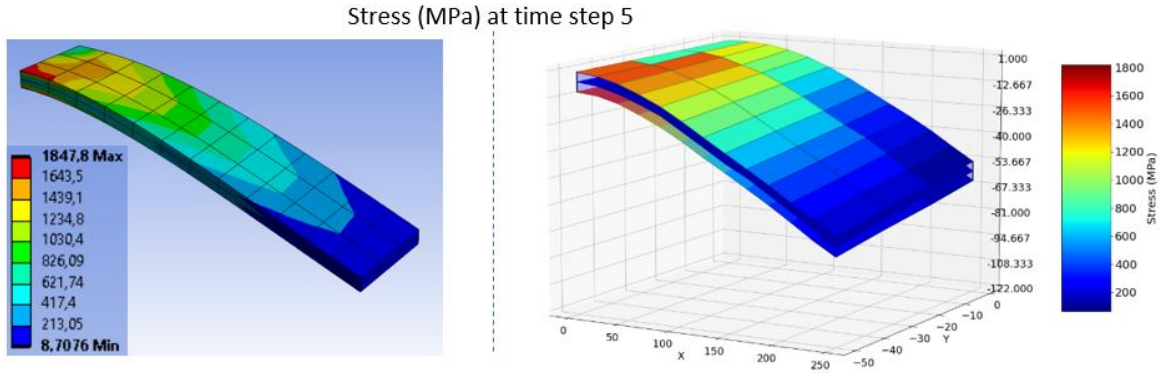

FIGURE 6.145: FEM vs. GG-NN result for stress at time step 5

It can be seen from figure 6.141 to 6.145 that although the GG-NN models rightly predicts the stress concentration regions on the cantilever beam, it fails on the part of value accuracy when we compare the colorbar values from FEA software result and predicted model result.

The actual value plots for nodal stress values can be seen in Appenndix section A.5.

Stress Error Visualization

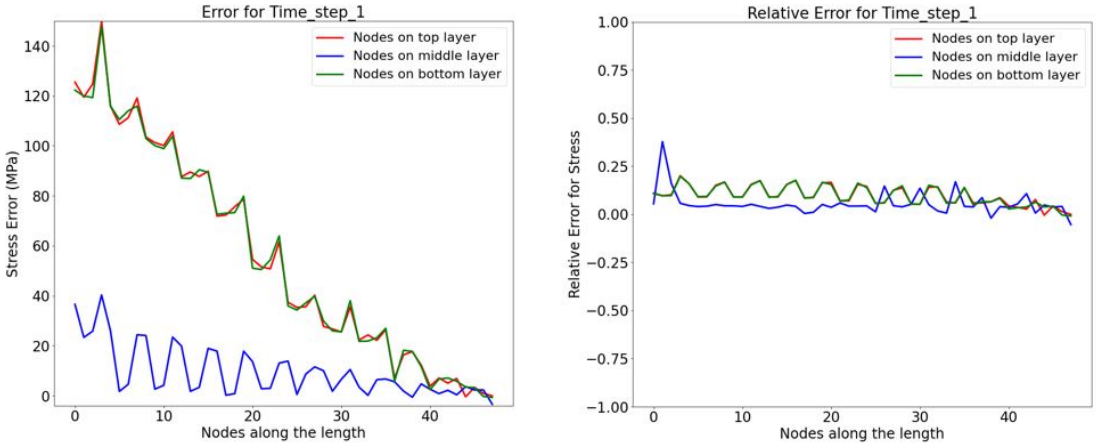

FIGURE 6.146: Error and relative error for stress at time step 1

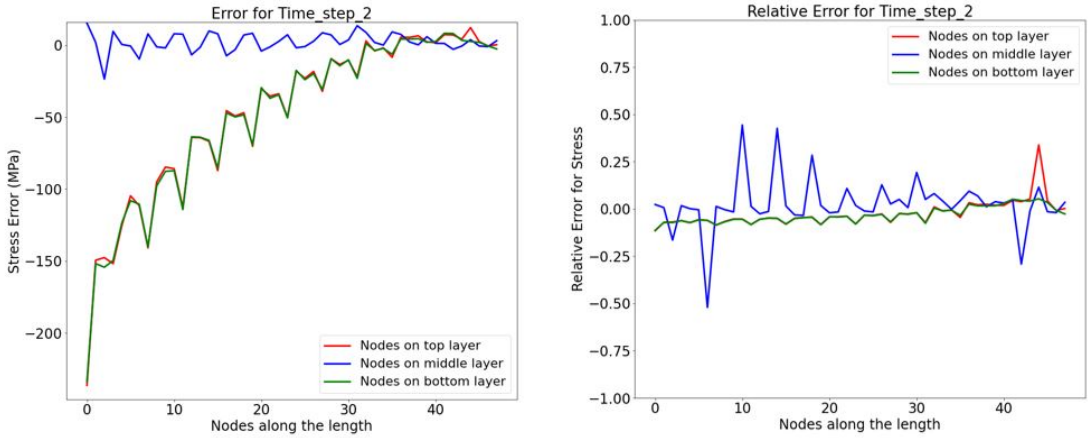

FIGURE 6.147: Error and relative error for stress at time step 2

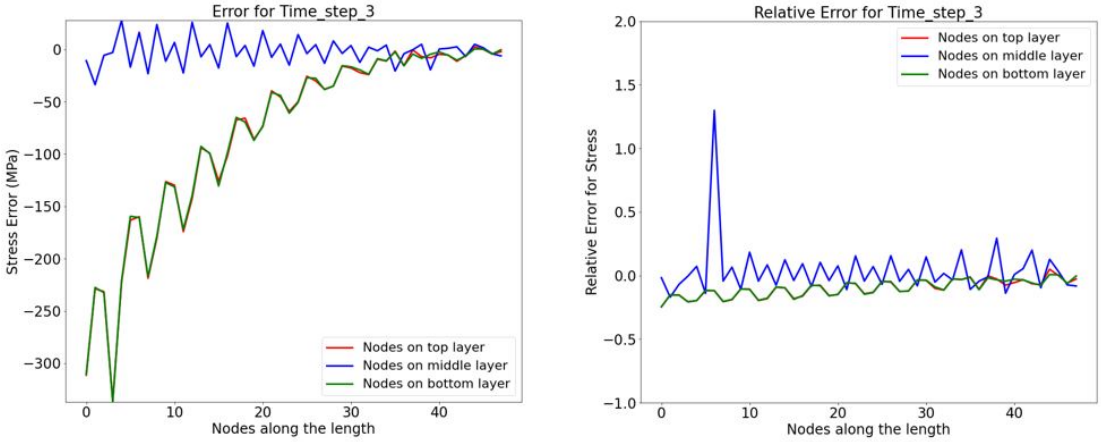

FIGURE 6.148: Error and relative error for stress at time step 3

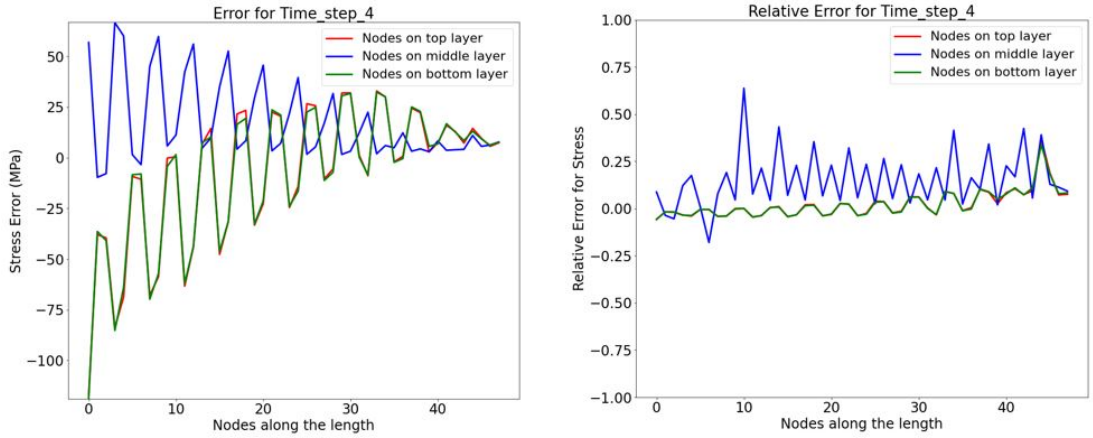

FIGURE 6.149: Error and relative error for stress at time step 4

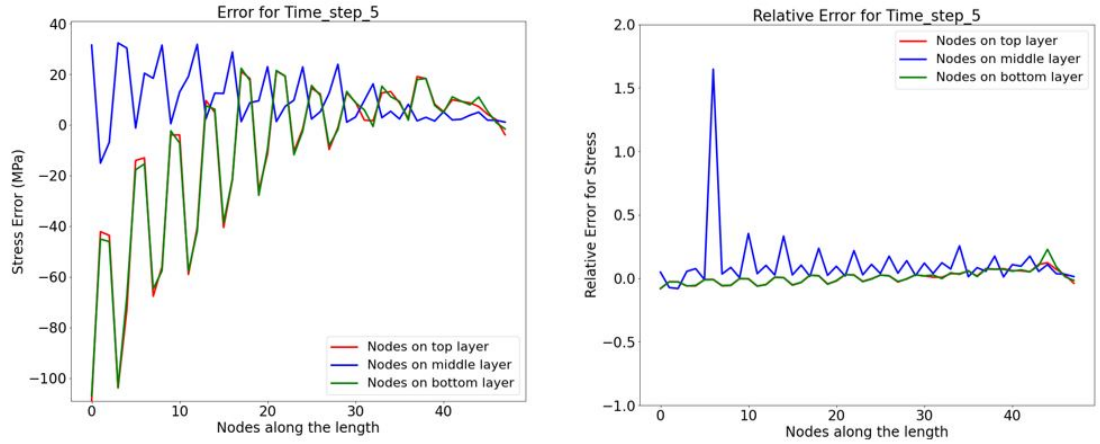

FIGURE 6.150: Error and relative error for stress at time step 5

From the plots in figure 6.146 to 6.150 it can be seen that for every time step the stress error exceeds 100 MPa. For time step 2 and 3 it even reaches maximum of 300 MPa which is of high significance when compared to the actual value and gets reflected in the relative error plot. It can thus be generalized that the GG-NN model gives unsatisfactory performance when it comes to stress value predictions.

6.4 Summary and Discussion

The main goal of the neural network models used and experimented with, is to give the best possible predictions for deformation and stress for given input force values. The three different experiments namely, the coarse mesh, finer mesh and the twisting case represent different variations in the simulation domain. Two different models with variation in graph networks namely, GCN and GG-NN in combination with MLP have been used to determine their effectiveness on the datasets for all three experiments. Their performance has been summarized in tables 6.4, 6.5 and 6.6.

Experiment Dataset	Model Architecture	Average Maximum Deformation Error at the Free End (mm)	Prediction Quality
Coarse Mesh	GCN+MLP	1.82	Good
	GG+MLP	1.63	Good
Finer Mesh	GCN+MLP	1.02	Good
	GG+MLP	1.08	Good
Twist Case	GCN+MLP	5.74	Satisfactory
	GG+MLP	14.90	Poor

TABLE 6.4: Model performances on deformation prediction

The deformation error at the free end or the tip of the cantilever indicates the maximum offset between the predicted beam deformed position and the actual beam deformed position. Hence an average of the maximum deformation error at the tip of the cantilever beam is taken from multiple mean results to denote the effectiveness of models. From table 6.4, it is observed that both the models performed exceptionally well in coarse mesh and finer mesh experiment. However in the twist case experiment the maximum deformation error was on the higher side. The GCN model gave satisfactory result while the model with GG-NN gave comparatively poor result.

Experiment Dataset	Model Architecture	Avgerage Maximum Stress Error near Fixed End (MPa)	Prediction Quality
Coarse Mesh	GCN+MLP	25.6	Good
	GG+MLP	146.6	Satisfactory
Finer Mesh	GCN+MLP	39.8	Good
	GG+MLP	78.0	Good
Twist Case	GCN+MLP	132.5	Satisfactory
	GG+MLP	197.0	Poor

TABLE 6.5: Model performances on stress prediction

From the 3D stress plots, it is observed that the maximum induced stress occurs in the region near the fixed end of cantilever beam. Hence the stress values obtained

in this region are critical from design and analysis perspective, for eg. it may help decide whether the material fails or fractures at that location. It is therefore important to have better stress predictions with least possible error at locations close to the fixed end. The average maximum stress error near the fixed end of cantilever beam from multiple mean results is thus taken into consideration to determine the effectiveness of the models. From table 6.5, it is observed that the model with GCN gave excellent stress prediction in coarse and finer mesh experiment and satisfactory performance in twist case experiment. The model with GG-NN gave good result in finer mesh experiment and satifactory result in coarse mesh experiment. It however performed poor in twist case experiment with high error.

Experiment Dataset	**Model Architecture**	**Overall Performance / Mean Absolute Error on Test Data**	**Best Model**
Coarse Mesh	GCN+MLP	4.02	GCN+MLP
	GG+MLP	16.53	
Finer Mesh	GCN+MLP	7.29	GCN+MLP
	GG+MLP	12.37	
Twist Case	GCN+MLP	8.81	GCN+MLP
	GG+MLP	26.03	

TABLE 6.6: Overall performance of models

From the results displayed in table 6.6, it can be summarized that the model with GCN gave the best results in all the three experiments. The GG-NN model performed well on coarse and finer mesh datasets but gave below par performance on twist case dataset. Some of the reasons for this could be- less number of observations in the twist case dataset (1400 compared to 2000 in coarse and finer mesh), insufficient input features for the model to learn, and the lack of suitability of information propagation method used in GG-NNs. Thus it can be concluded that the model with GCN is the safest and the best performance choice.

Chapter 7

Conclusion and Future Scope

The primary objective of this thesis is to develop a Machine Learning approach using Deep Neural Network architectures that would replicate the results of the Finite Element Method based simulation problem. Simulations play a key role in areas of design and manufacturing. While there are various software platforms providing FE simulations, the limitations remain in form of time required for complex simulations and rerunning the entire simulation for even a small change in inputs. This could greatly affect today's fast paced production environment in terms of time and cost. The data-driven ML based approach is thus suitable solution in these scenarios. This thesis contributes to the solution as it displays the development of DNN model for predicting the output of simulation scenarios on a cantilever beam as a case study. Two DNN models, one with GCN architecture and other with GG-NN architecture were derived and tested on datasets from three different simulation scenarios of cantilever beam. Both the models gave good performance on majority of experiment cases confirming to the actual values obtained from the FEA software.

The input to the network model is in form of sequence of force values at five different time steps and the output is the corresponding deformation and stress sequence. The aim was to train the model such that for a given input force setting, it gives the output deformation and stress values confirming to the values from FEA with least possible error. The Mean Absolute Error (MAE) has been used as a measure to observe and compare the model performance. Besides this measure, the 3D plots using matplotlib give better visualization of results.

The GCN model and GG-NN model were tested in three different experiments. The first experiment represents the initial phase in any simulation task which is to get an initial intuition about the possible behaviour of a part or component (in this case a cantilever beam) by observing the results obtained on coarse mesh setting. Both the models were trained on the dataset obtained for the coarse mesh setting and have shown great results when compared with the actual values from the FEA software. The second experiment represents the next step in simulation process which is increasing the reliability of the results (getting more accurate output values) by increasing the number of elements and nodes or in other words- making the mesh finer. Again, both the models were trained and tuned on the finer mesh dataset and showed remarkable results. These two experiments dealt with single deformation output in downward Z direction for bending of cantilever beam. The third experiments goes a step further for predicting additional X and Y values for deformation by making the cantilever beam twist along with the bend. While the GCN model gave good results, the performance of GG-NN model was below par

with high error in deformation and stress predictions making the GCN model the best and reliable architecture for replacing FEM based FE simulations.

Apart from the prediction accuracy, the other important advantage of the developed neural network models over software based FE simulation is the required computation time. While it took around 48 hours to generate simulation data of 300 observation rows, the developed neural network models trained and gave results for 300 test data observations in about 25 minutes. Based on the results obtained, the developed architectures definitely have a good prospect for industrial applications provided the availability of past historic simulation data or the capacity to facilitate the storing of new generated data.

Although this thesis covers the variation in meshing and deformation output, there are other aspects like change in the geometry or the material properties that can be experimented with in future. Including these parameters in training will broaden the model predictions further making it more universal. The output conclusions can be expanded by including classification tasks to determine if a component fails or not for given inputs. Also, the GNNs itself being recently introduced, a lot of variations are possible in the methods of aggregating and propagating the information through the layers. These aspects can be definitely useful for a certain application specific scenario. Experimenting with other already available graph architectures for simulation specific tasks further opens up wide exploring possibilities. It would also be interesting to see the performance of graph network architectures in other simulation domains like modal analysis and thermal analysis. Efforts need to be made to accommodate and integrate the DNNs in the field of simulations further. Another interesting prospect would be the system integration of FEA simulations with the network model for output feedback and data updation in form of knowledge bank for the neural networks. With all these possibilities, the DNNs surely have the potential to elevate the current simulation methods to the next level.

Appendix A

Appendix A - Nodal Deformation and Stress Plots

A.1 Gated Graph Architecture on Coarse Mesh

Plots for Nodal Deformation Values

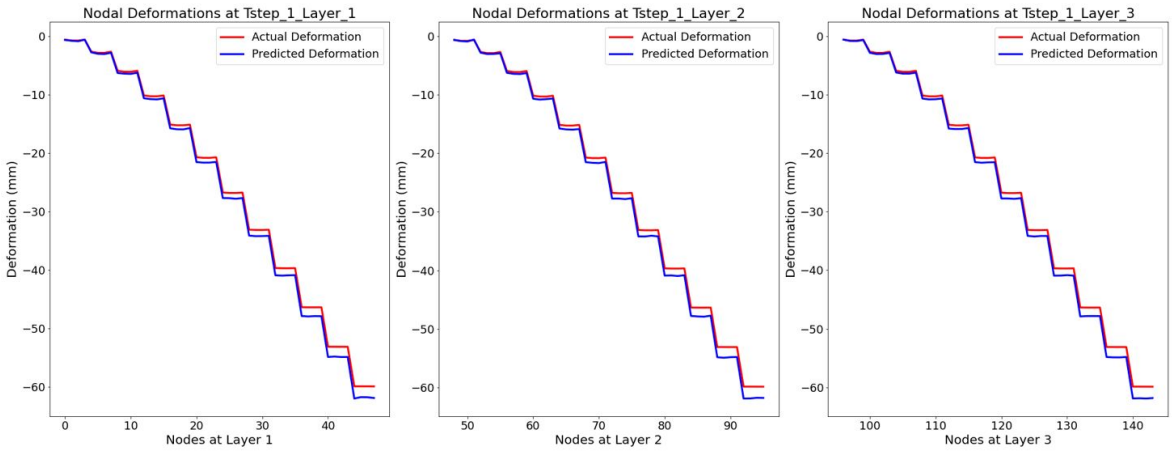

FIGURE A.1: Nodal deformations for time step 1

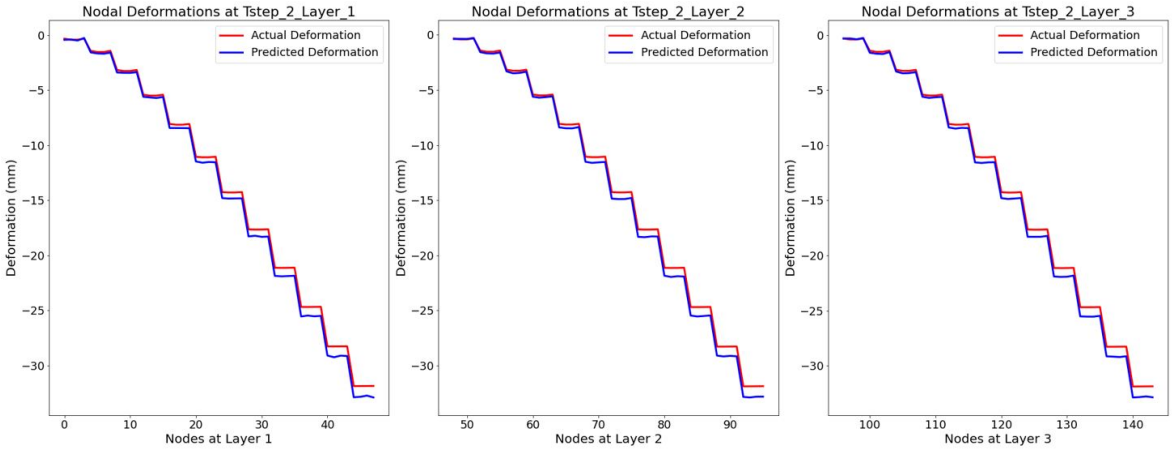

FIGURE A.2: Nodal deformations for time step 2

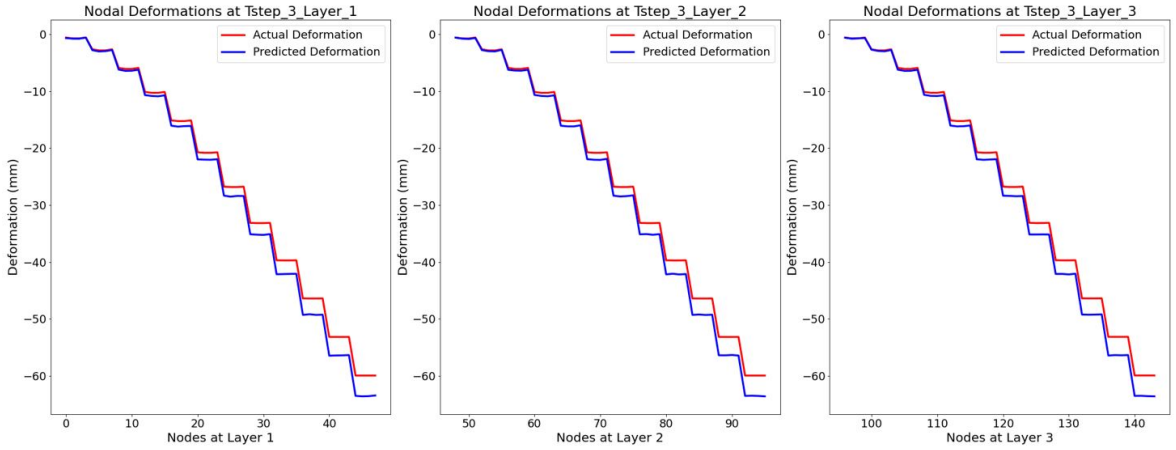

FIGURE A.3: Nodal deformations for time step 3

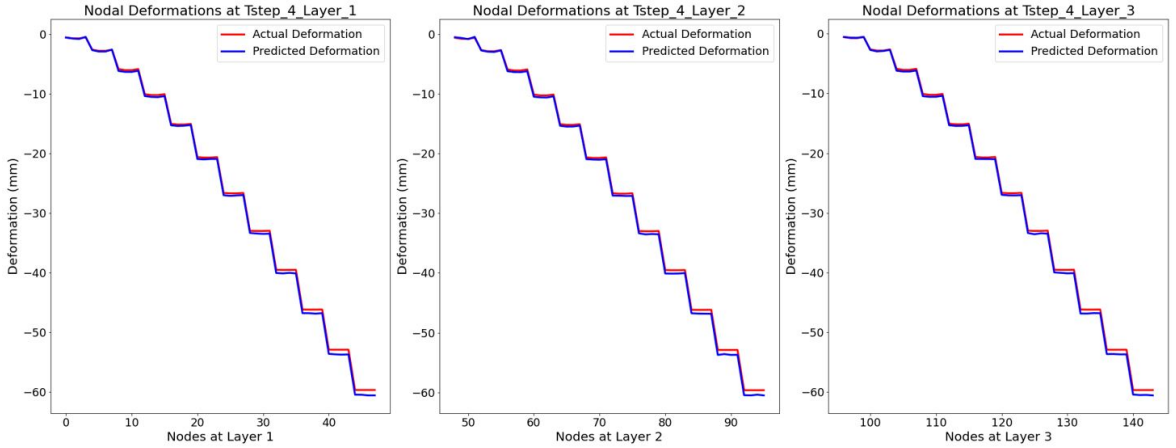

FIGURE A.4: Nodal deformations for time step 4

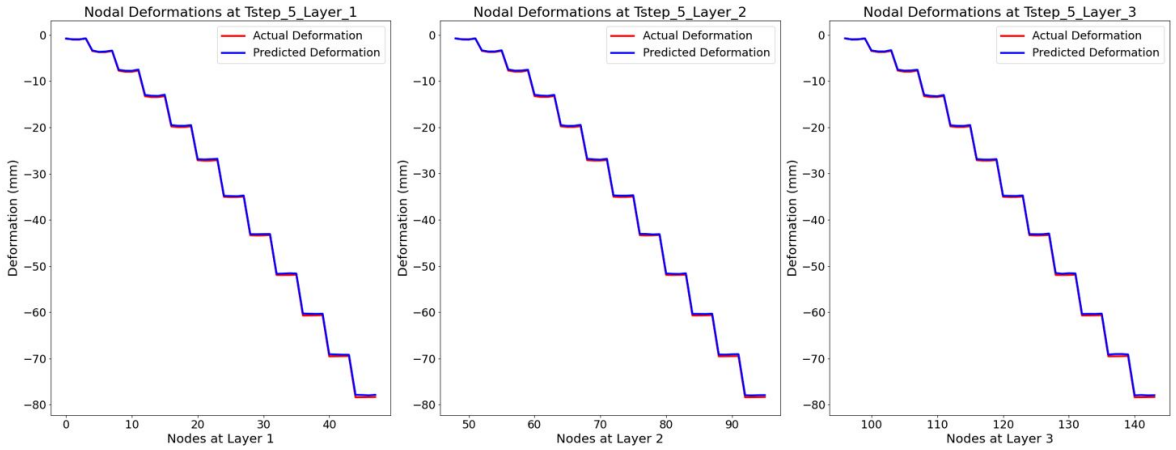

FIGURE A.5: Nodal deformations for time step 5

The plots from figure A.1 to A.5 show slight deviation in predicted nodal deformations for time steps 1, 2 and 3. Overall the plots show good prediction quality as the error is small (upto 4mm).

Plots for Nodal Stress Values

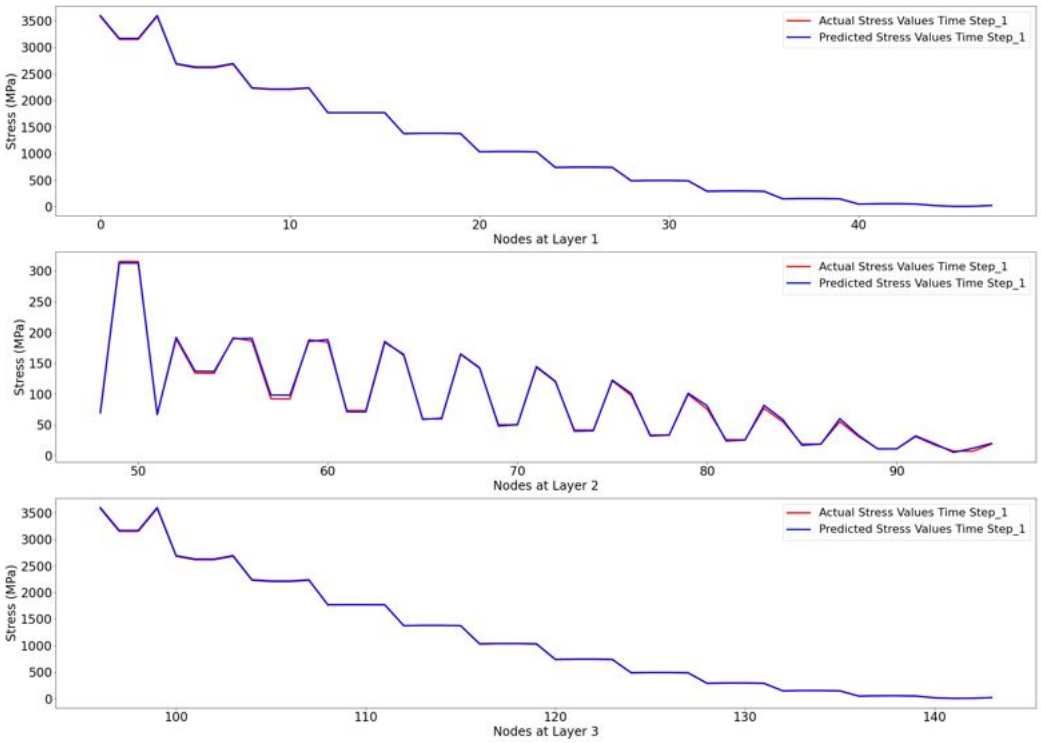

FIGURE A.6: Nodal stresses for time step 1

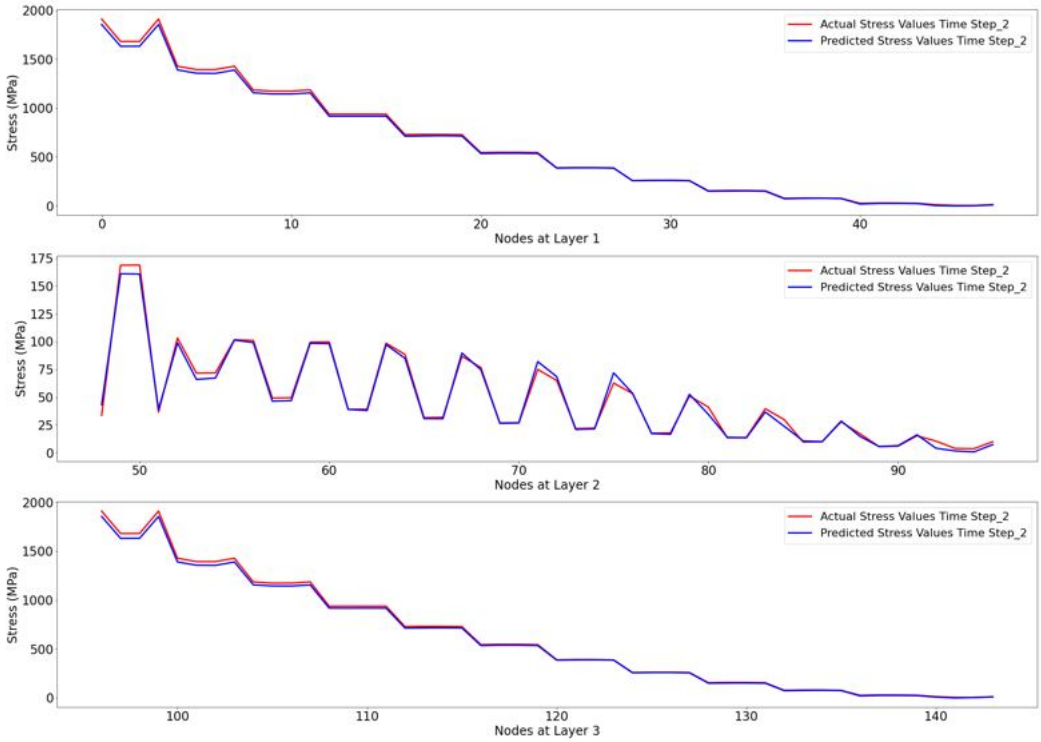

FIGURE A.7: Nodal stresses for time step 2

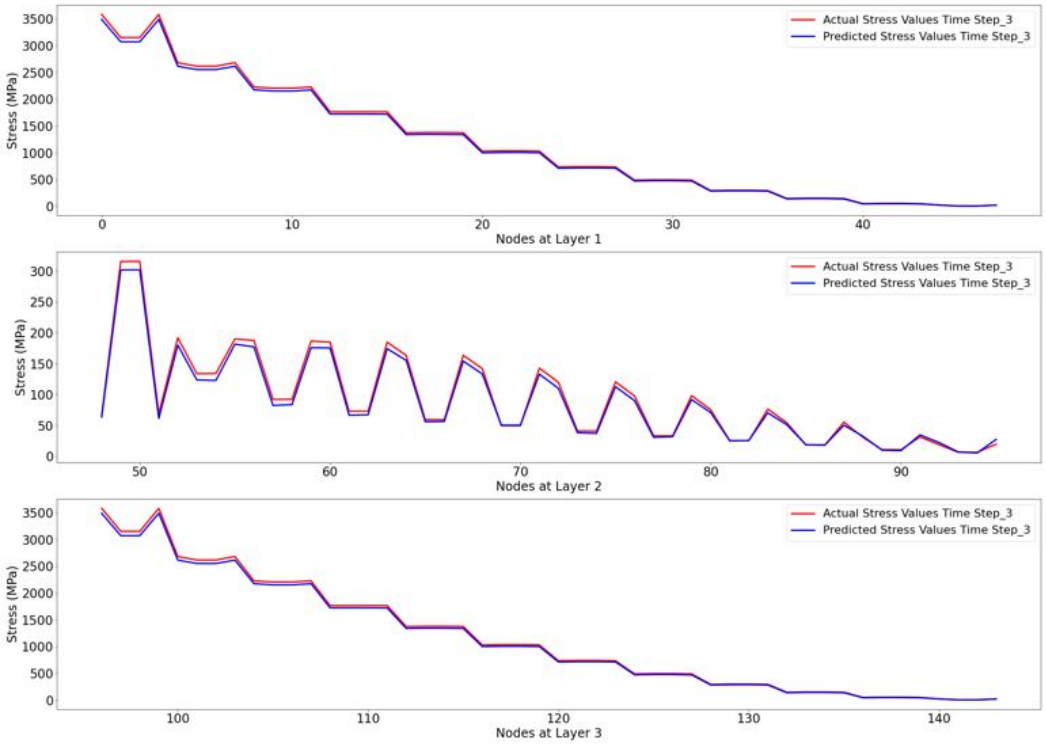

FIGURE A.8: Nodal stresses for time step 3

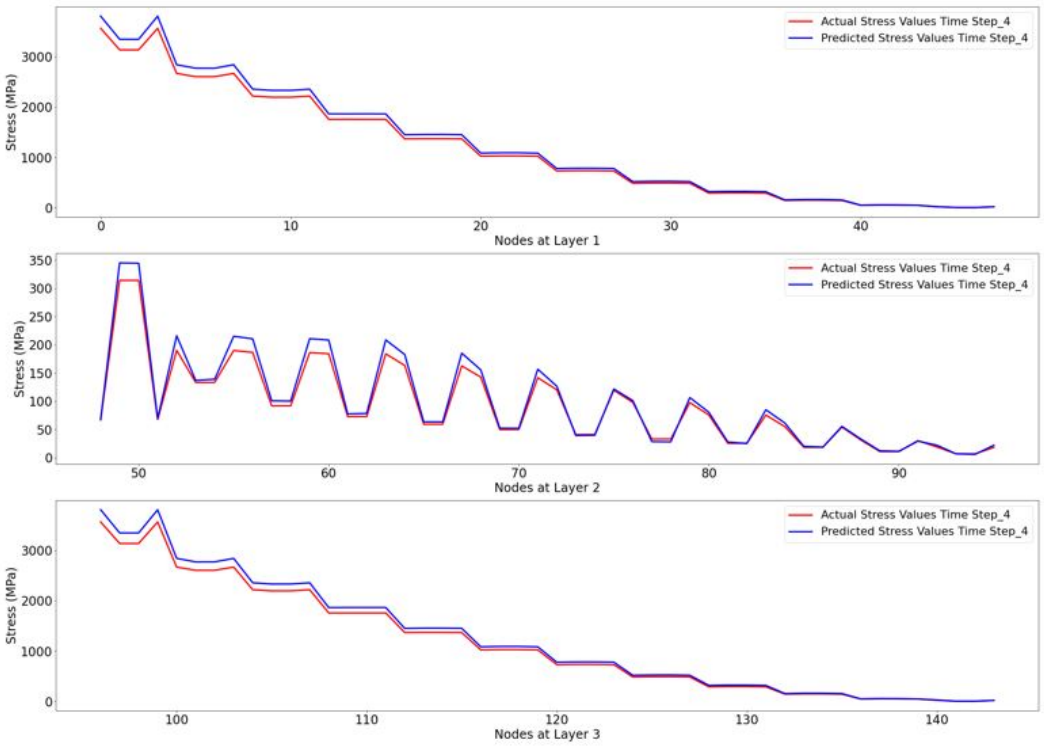

FIGURE A.9: Nodal stresses for time step 4

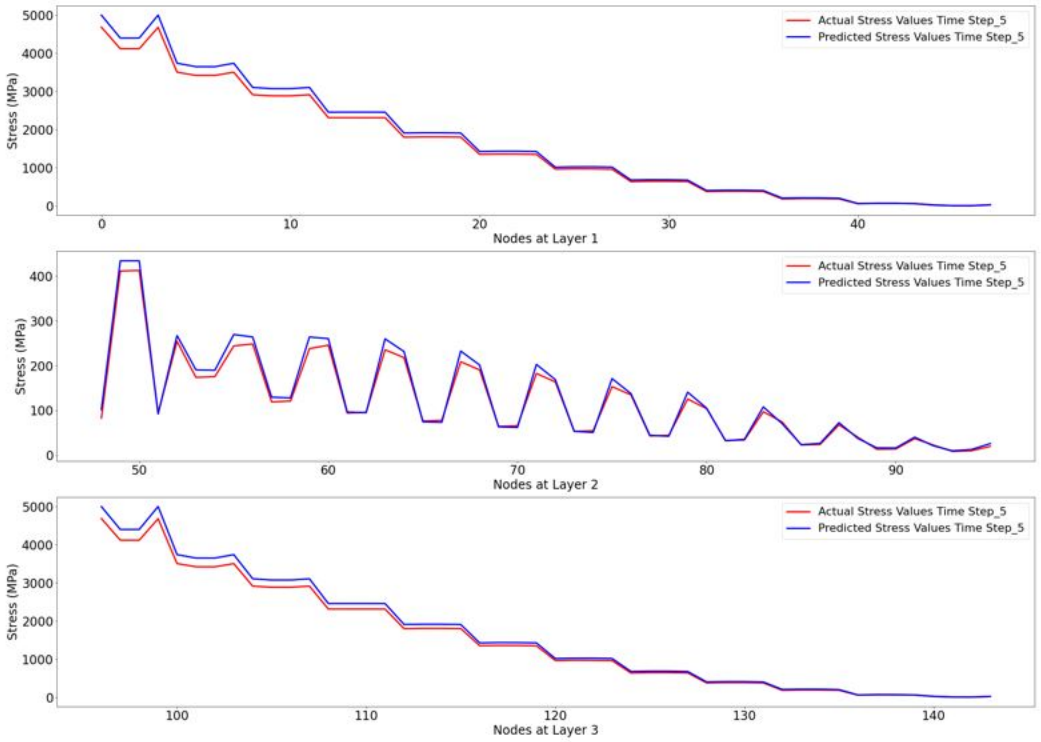

FIGURE A.10: Nodal stresses for time step 5

The nodal stress predictions as seen in figure A.6 to figure A.10 indicate some difference in actual and predicted values for time step 4 and 5. The overall performance is satisfactory.

A.2 GCN Architecture on Finer Mesh

Plots for Nodal Deformation Values

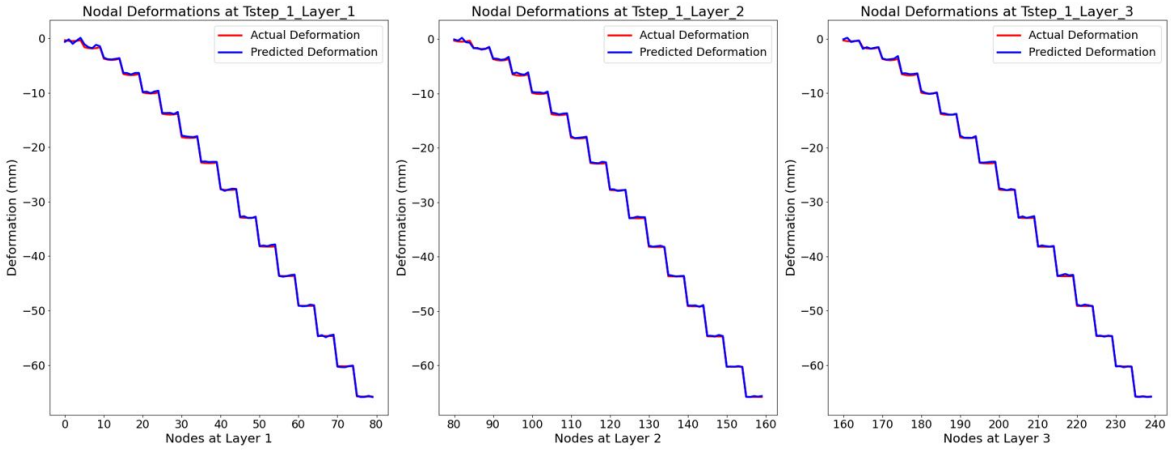

FIGURE A.11: Nodal deformations for time step 1

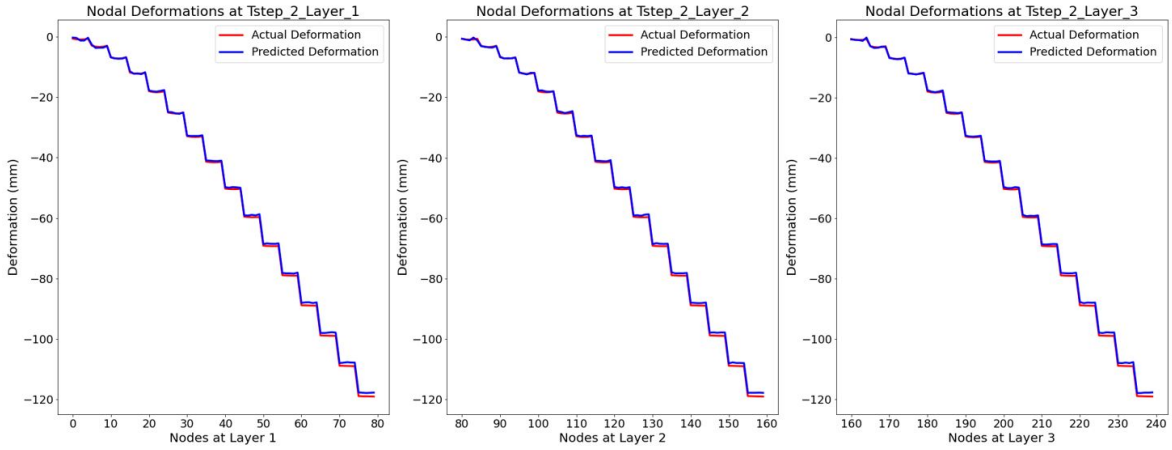

FIGURE A.12: Nodal deformations for time step 2

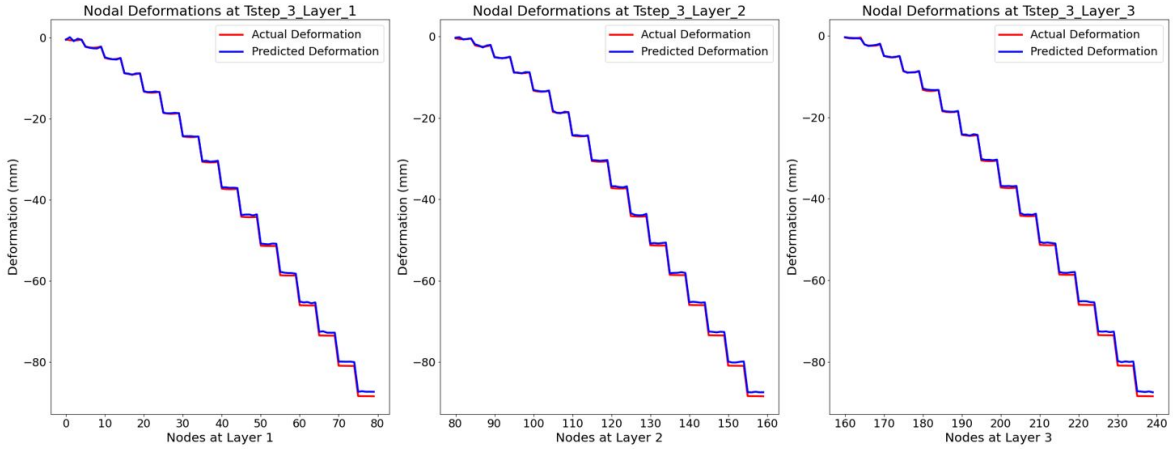

FIGURE A.13: Nodal deformations for time step 3

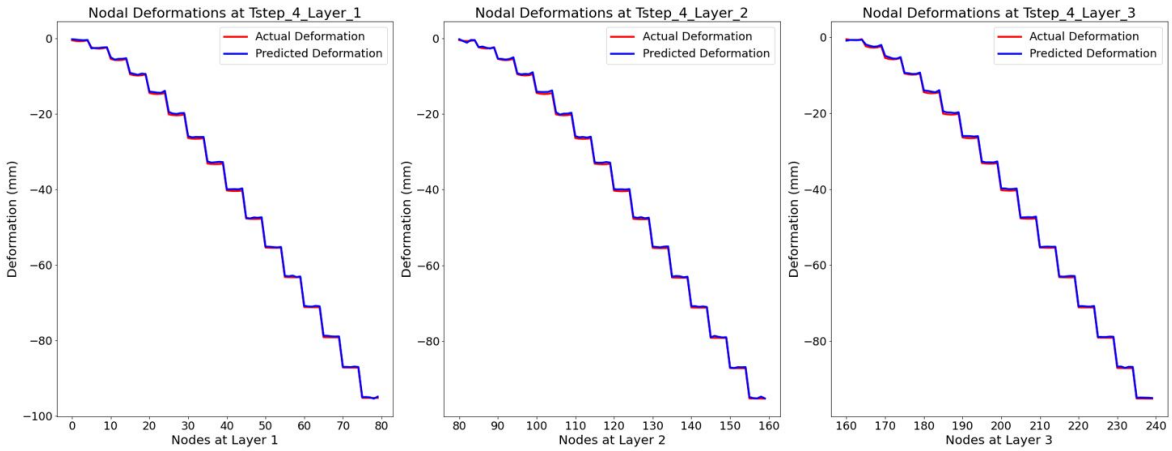

FIGURE A.14: Nodal deformations for time step 4

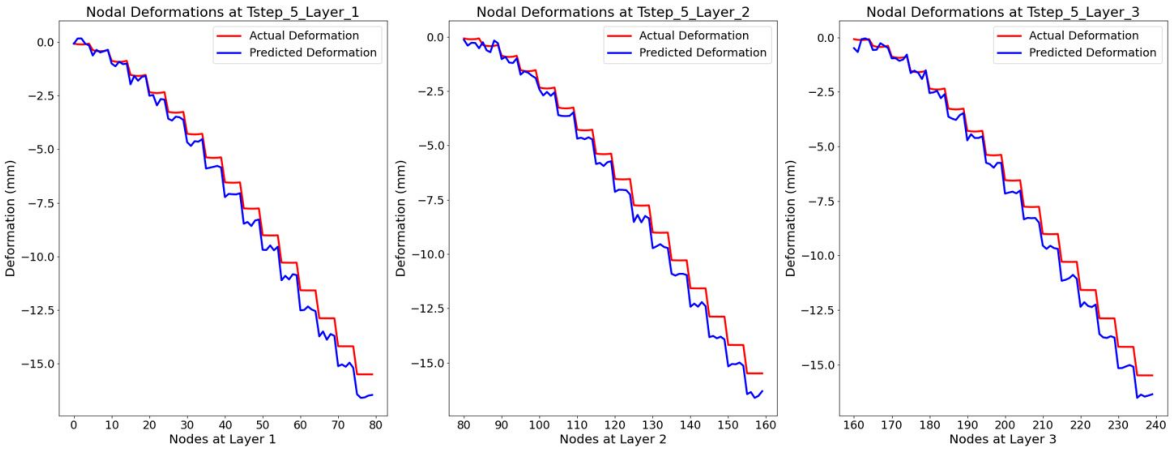

FIGURE A.15: Nodal deformations for time step 5

It can be seen that all the time steps have predicted nodal deformations line overlapping with actual nodal deformation line. A slight deviation of 1mm is observed in the final time step.

Plots for Nodal Stress Values

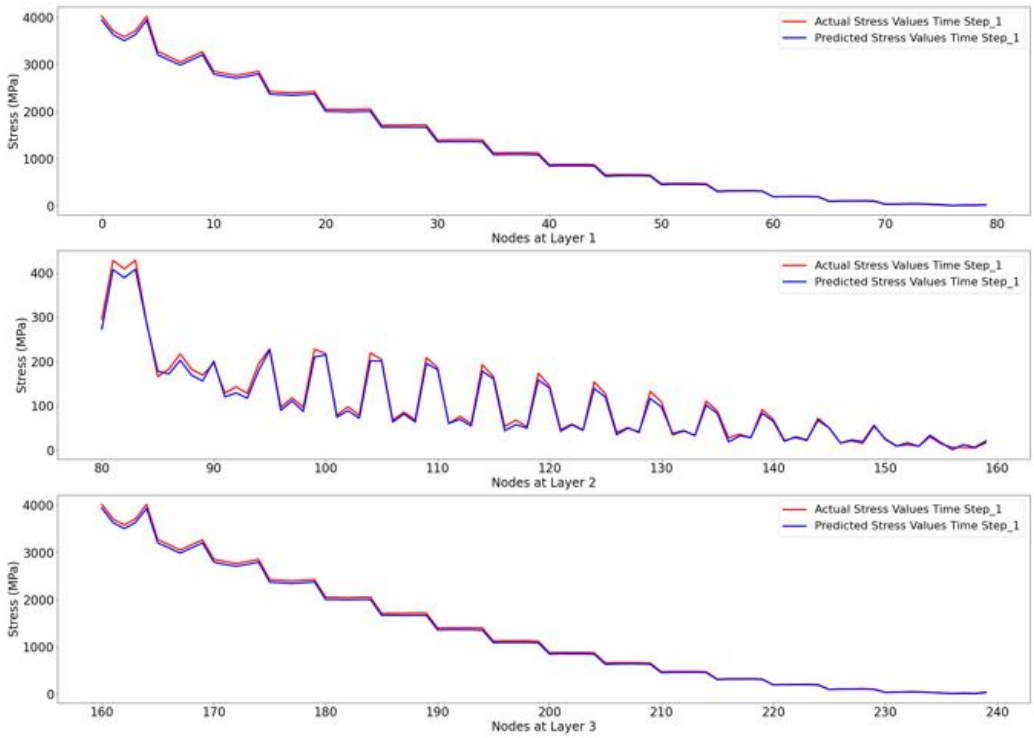

FIGURE A.16: Nodal stresses for time step 1

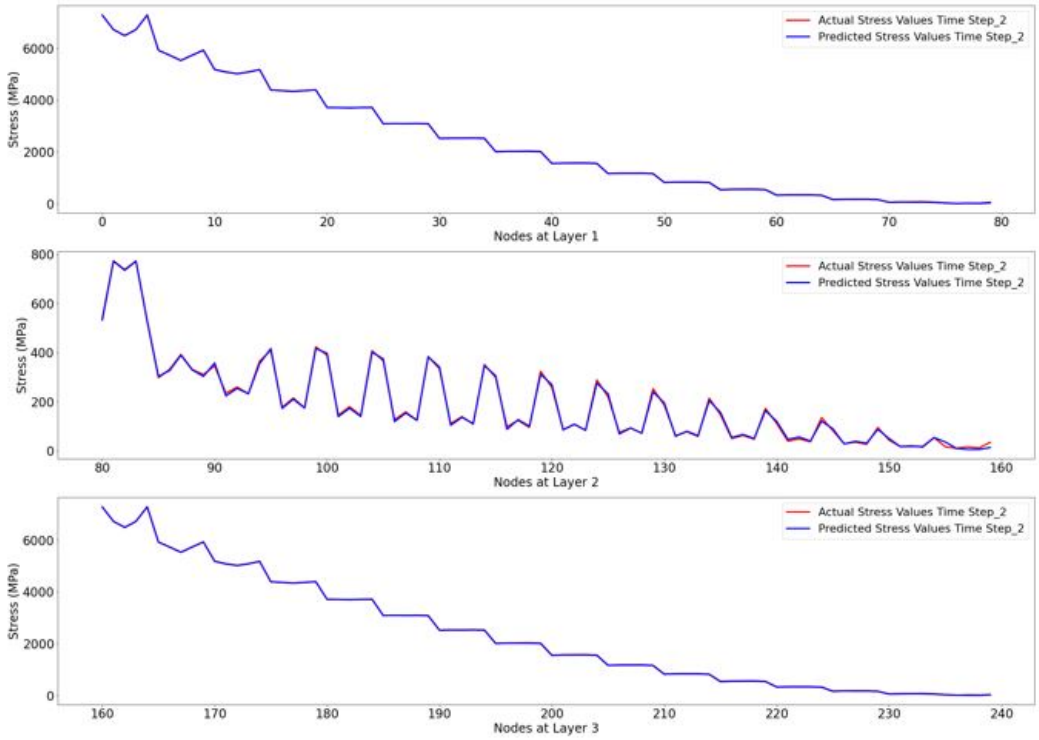

FIGURE A.17: Nodal stresses for time step 2

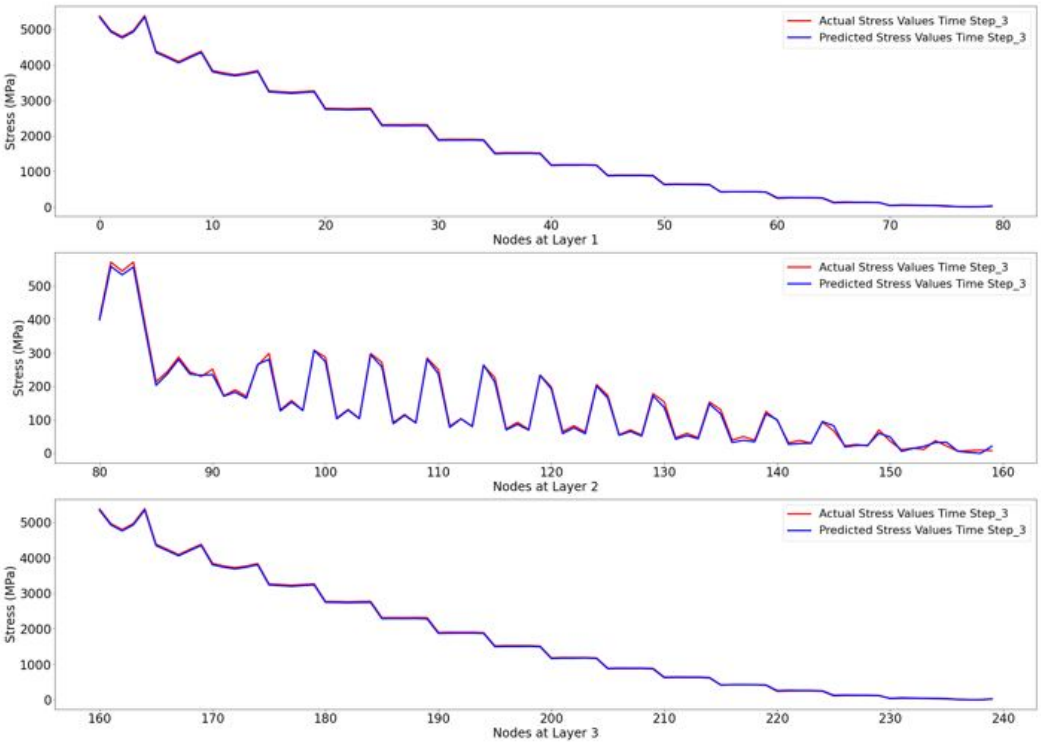

FIGURE A.18: Nodal stresses for time step 3

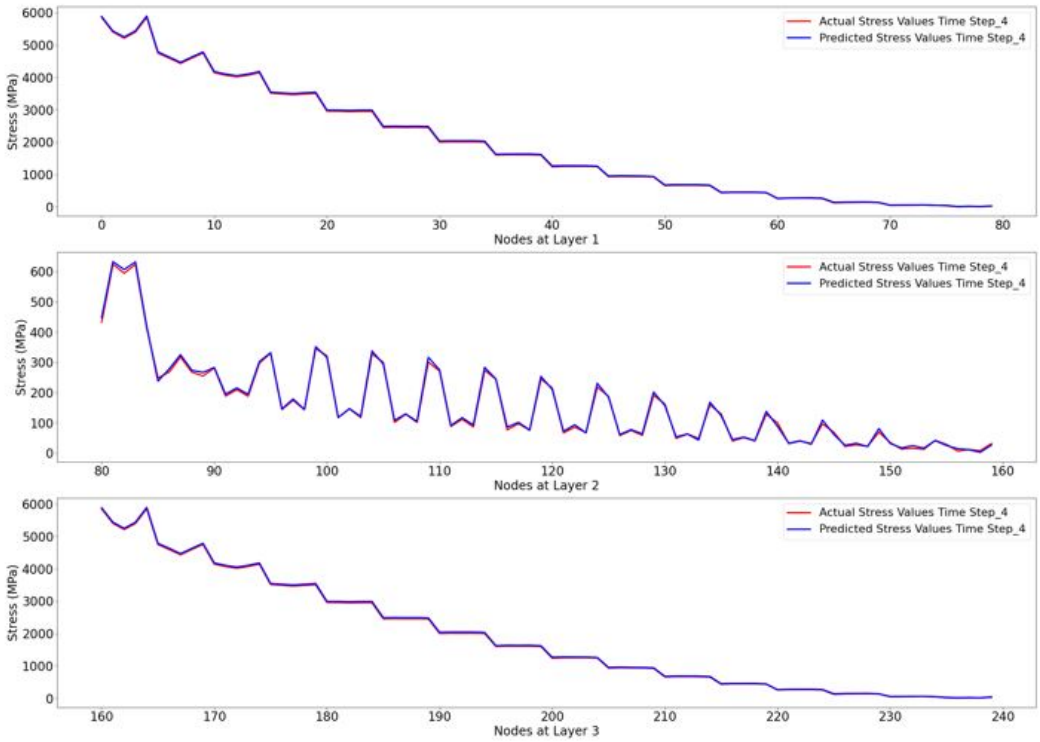

FIGURE A.19: Nodal stresses for time step 4

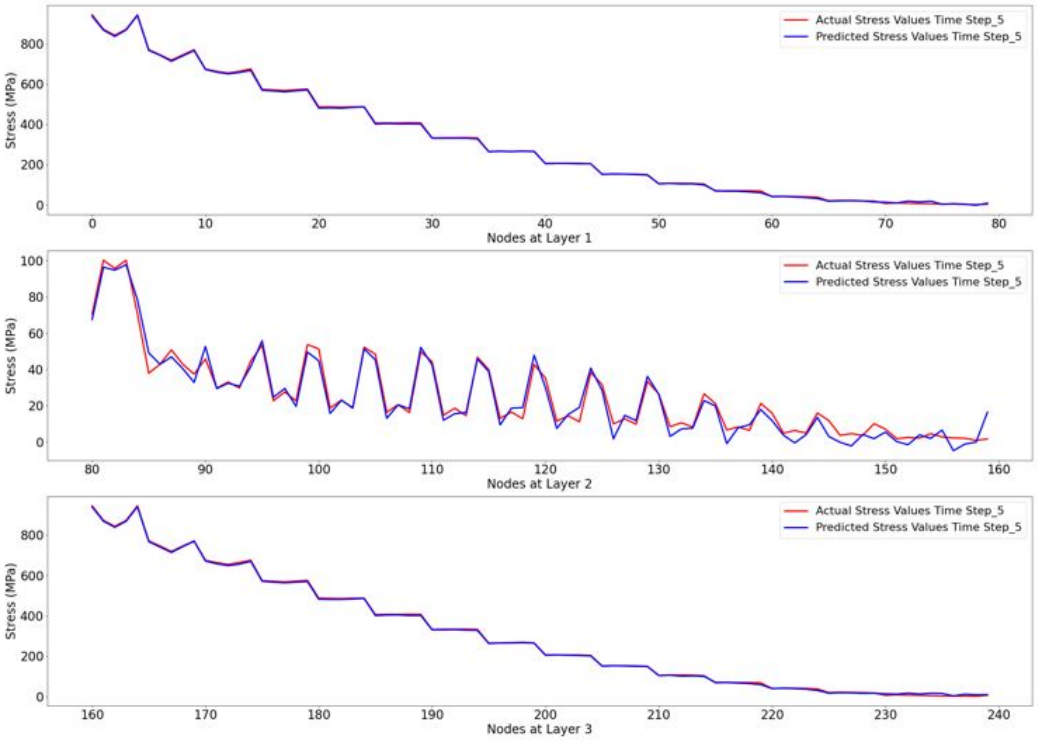

FIGURE A.20: Nodal stresses for time step 5

The overlapping actual stress value lines and predicted stress value lines for the nodes observed in plots from figure A.16 to A.20 show excellent model predictions.

A.3 Gated Graph Architecture on Finer Mesh

Plots for Nodal Deformation Values

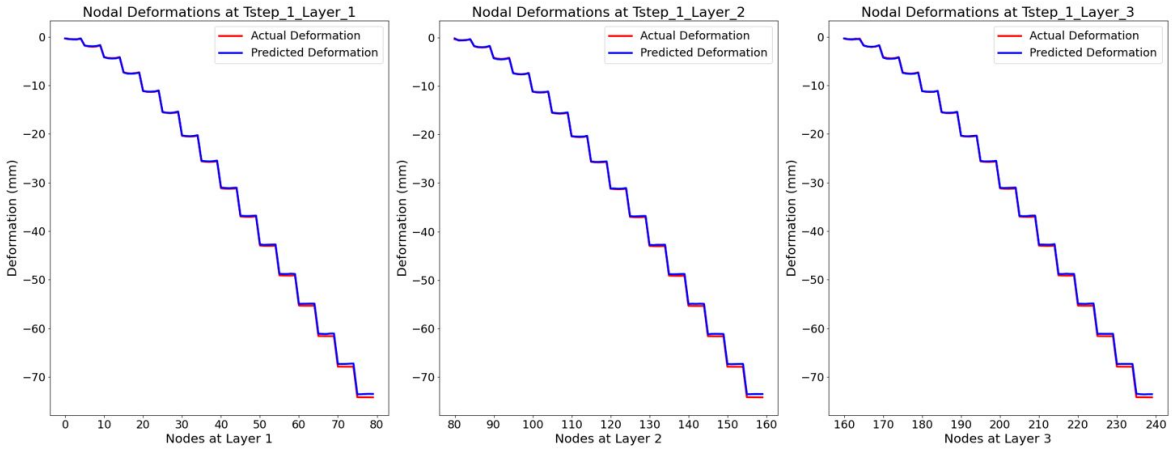

FIGURE A.21: Nodal deformations for time step 1

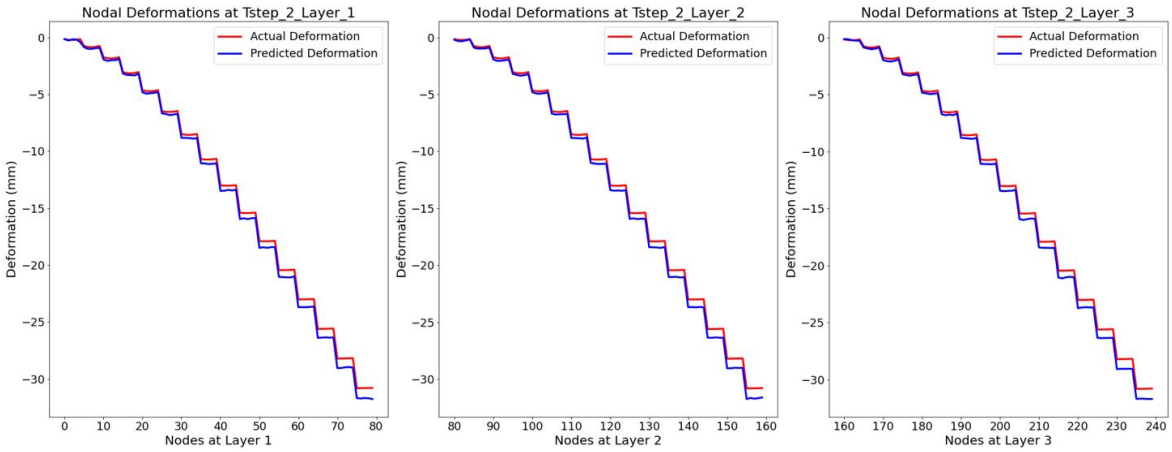

FIGURE A.22: Nodal deformations for time step 2

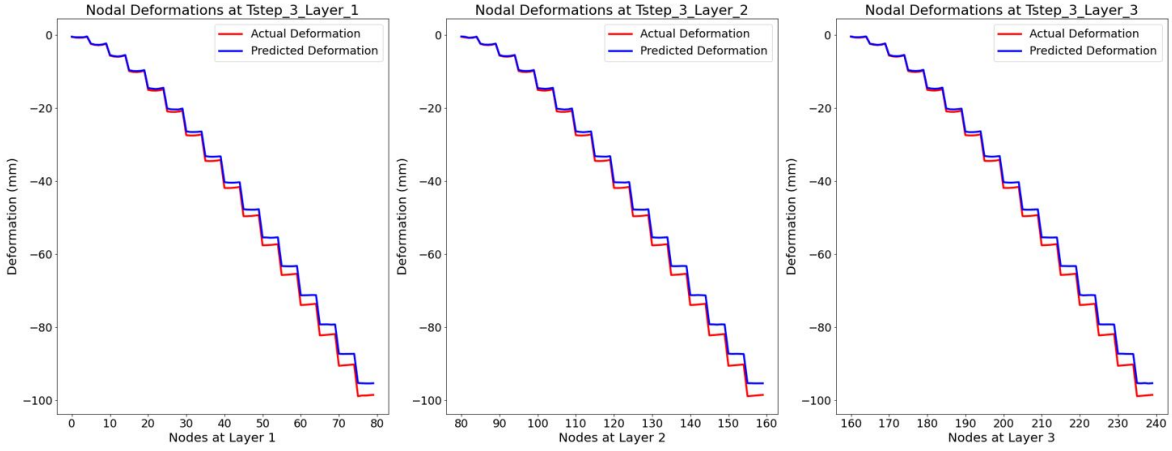

FIGURE A.23: Nodal deformations for time step 3

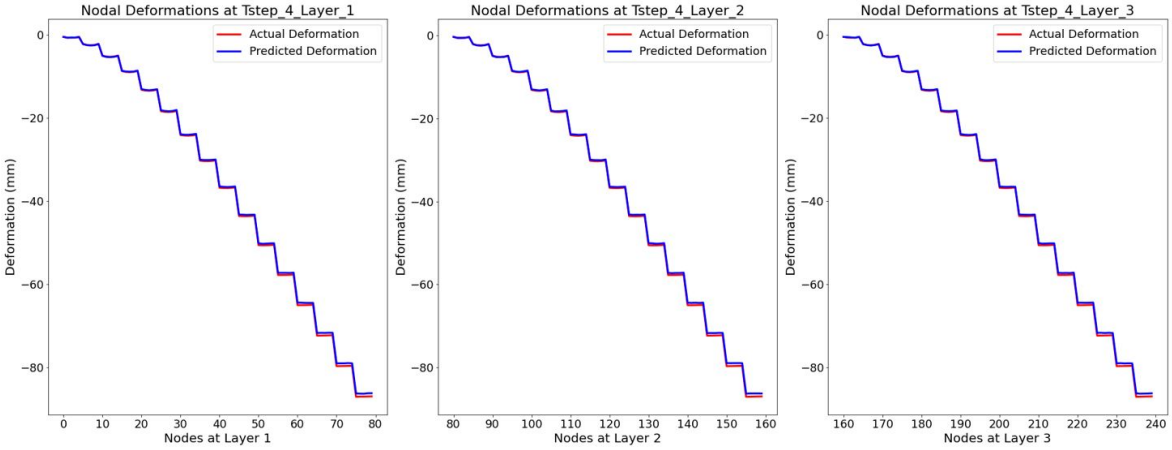

FIGURE A.24: Nodal deformations for time step 4

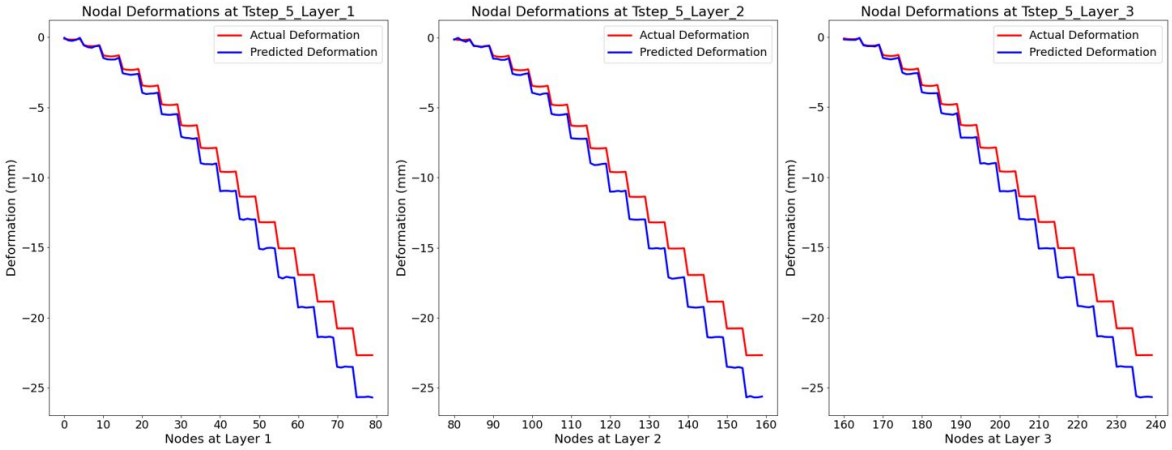

FIGURE A.25: Nodal deformations for time step 5

From figure A.21 to A.25 it can be stated that except for time step 5 the predicted nodal deformations values confirm to the actual values. The GG-NN model therefore is capable of giving good deformation predictions.

Plots for Nodal Stress Values

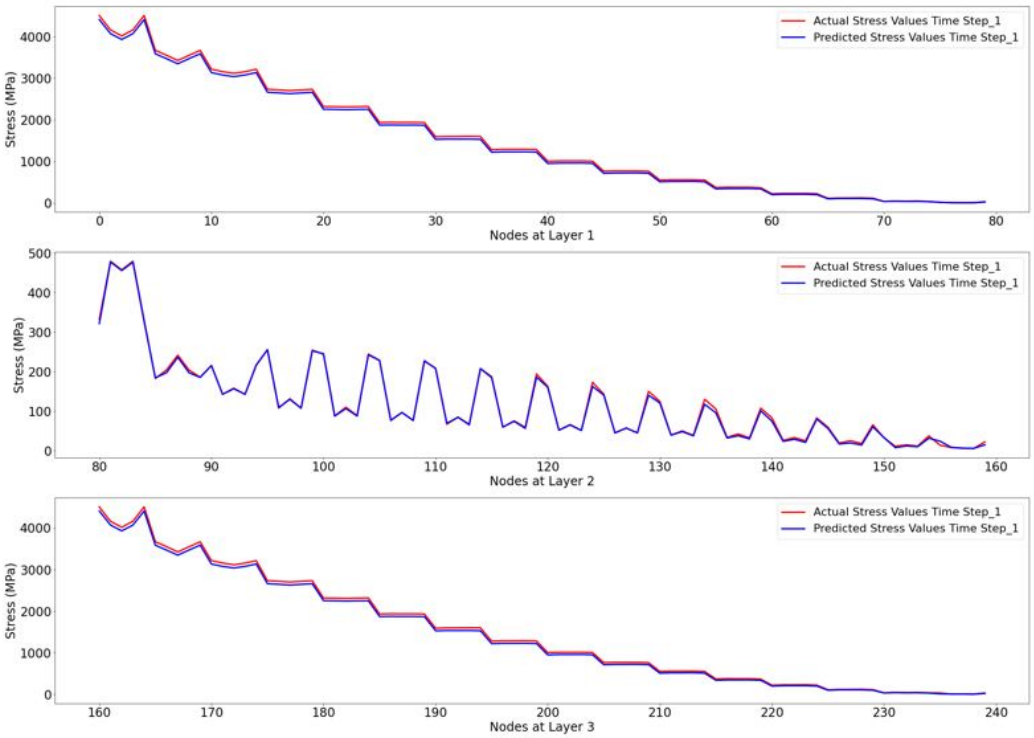

FIGURE A.26: Nodal stresses for time step 1

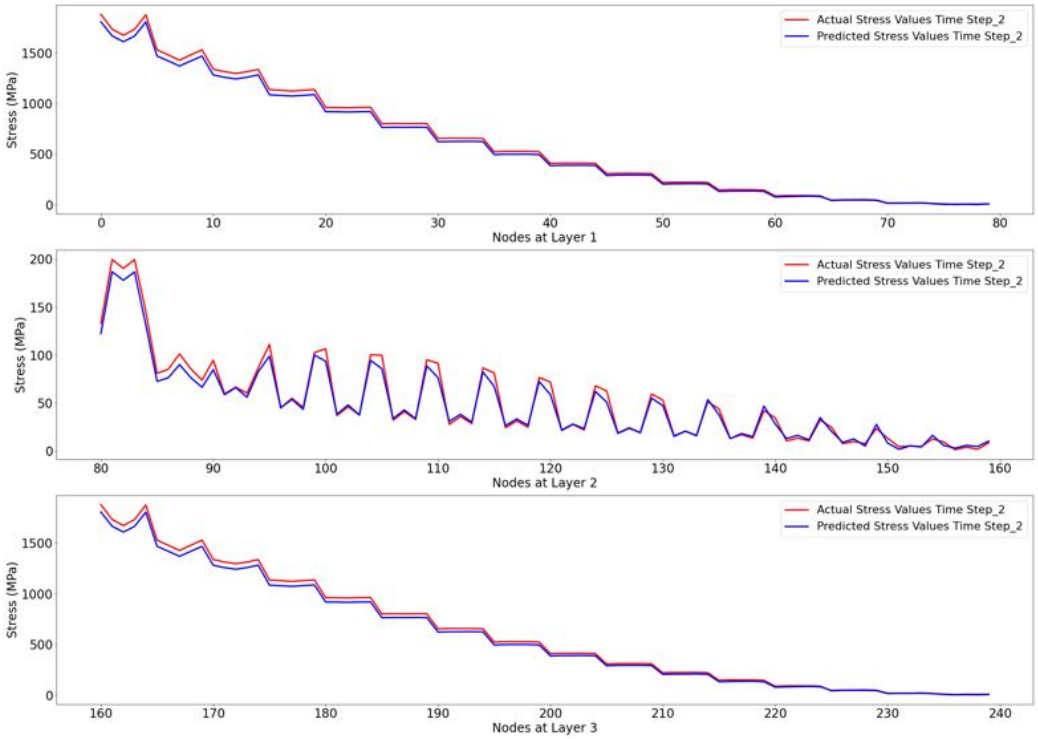

FIGURE A.27: Nodal stresses for time step 2

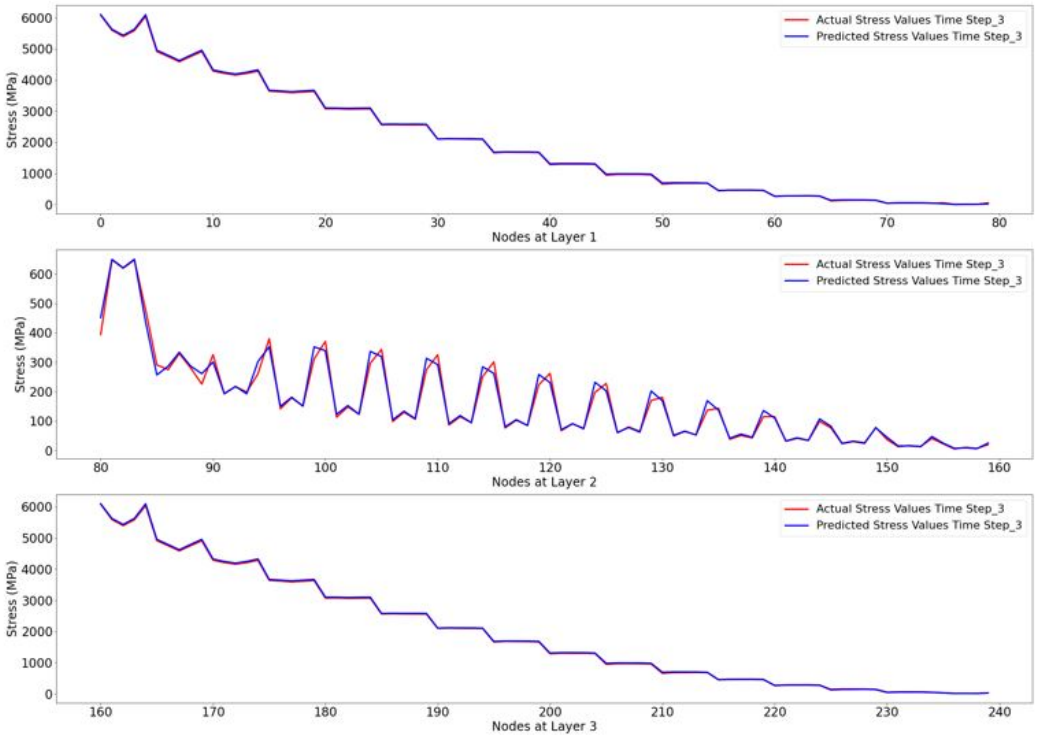

FIGURE A.28: Nodal stresses for time step 3

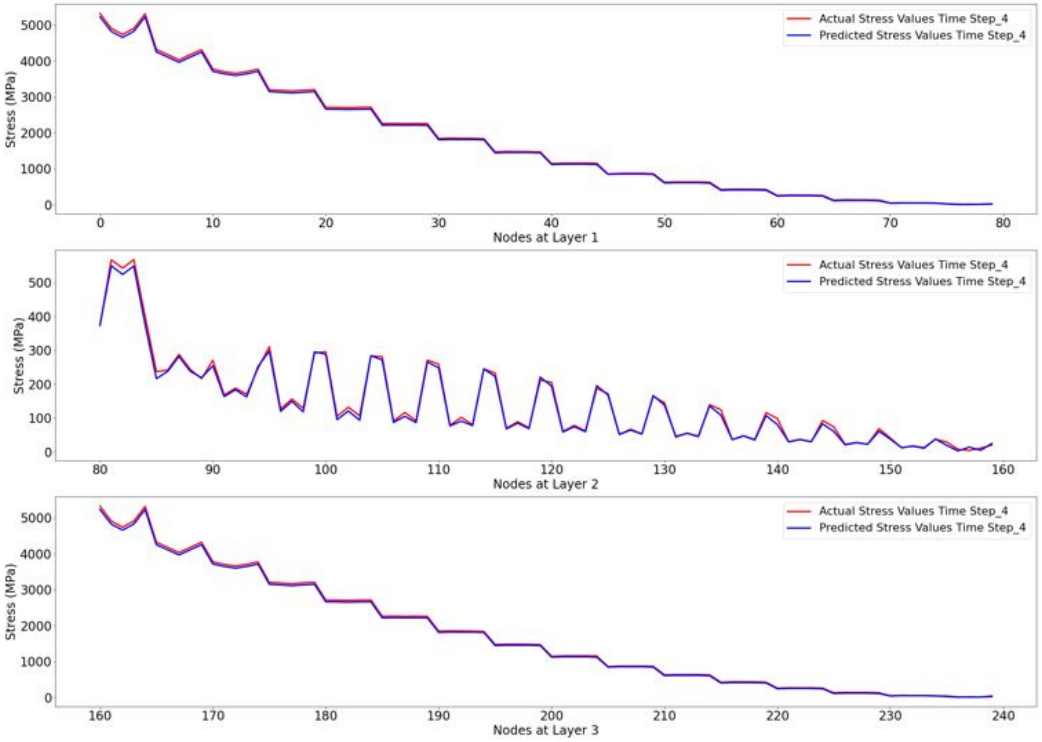

FIGURE A.29: Nodal stresses for time step 4

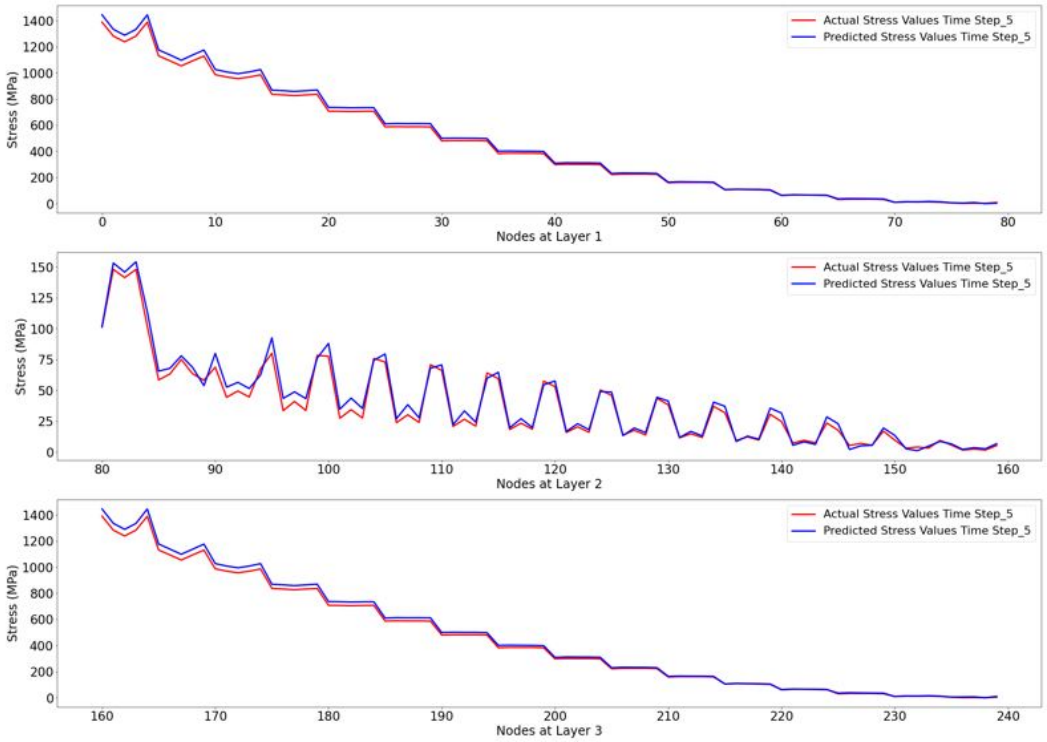

FIGURE A.30: Nodal stresses for time step 5

The overlapping nature of actual stress and predicted stress lines from figure A.26 to A.30 show thst the GG-NN model is capable of giving good stress predictions.

A.4 GCN Architecture on Twist Case

Plots for Nodal Z Deformation Values

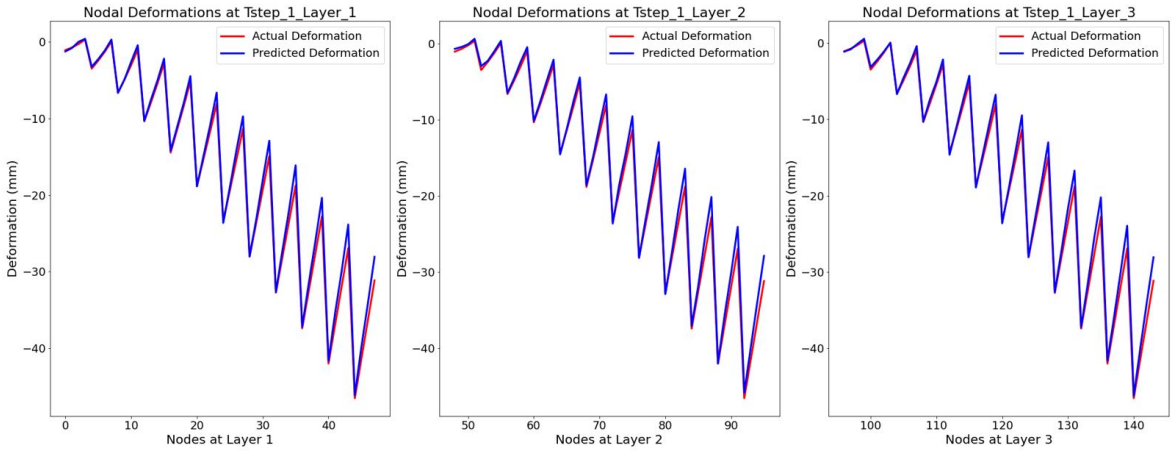

FIGURE A.31: Nodal Z deformations for time step 1

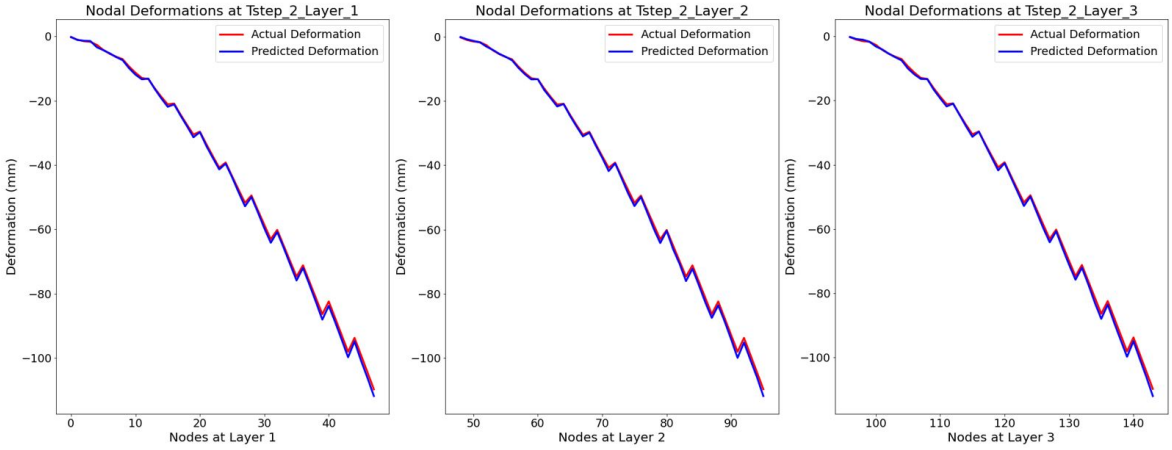

FIGURE A.32: Nodal Z deformations for time step 2

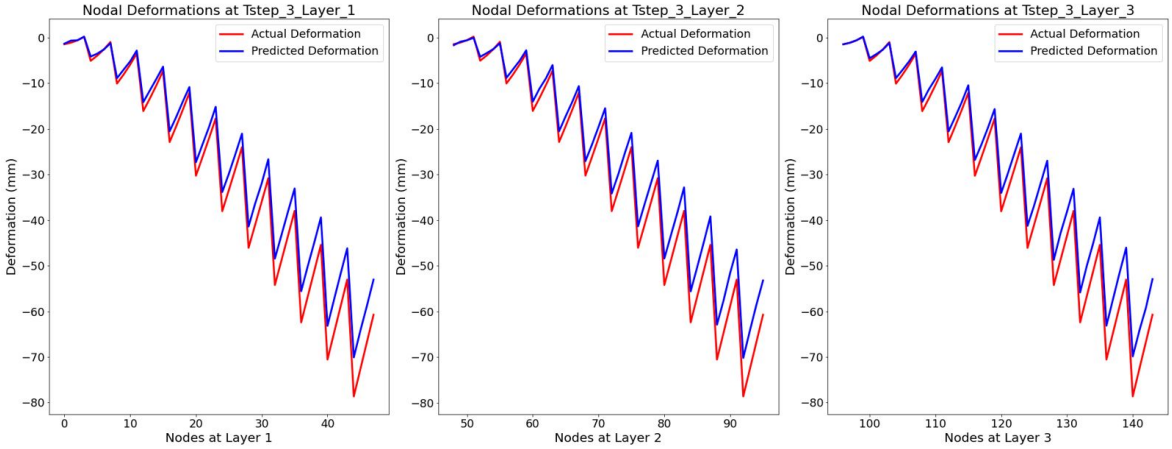

FIGURE A.33: Nodal Z deformations for time step 3

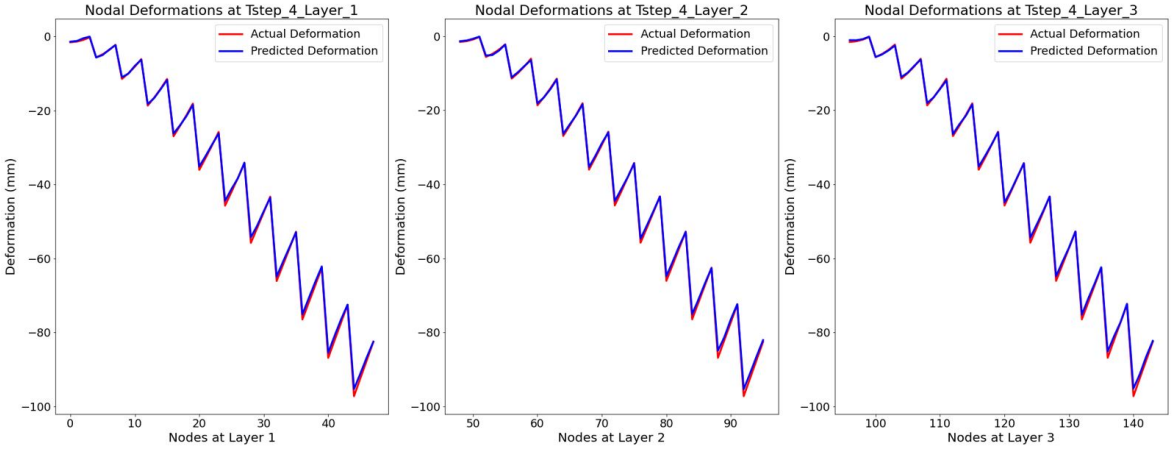

FIGURE A.34: Nodal Z deformations for time step 4

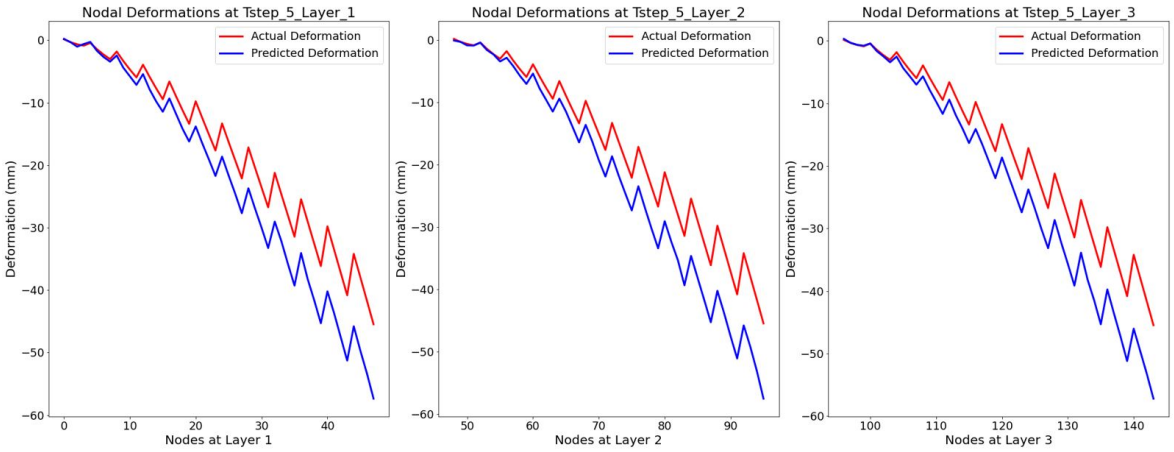

FIGURE A.35: Nodal Z deformations for time step 5

Apart from time step 3 and 5 in figure A.33 and A.35 respectively, the rest of the plots confirm the overlapping nature of actual and predicted Z deformation values for the nodal points. This confirms satisfactory performance of the GCN model.

Plots for Nodal Stress Values

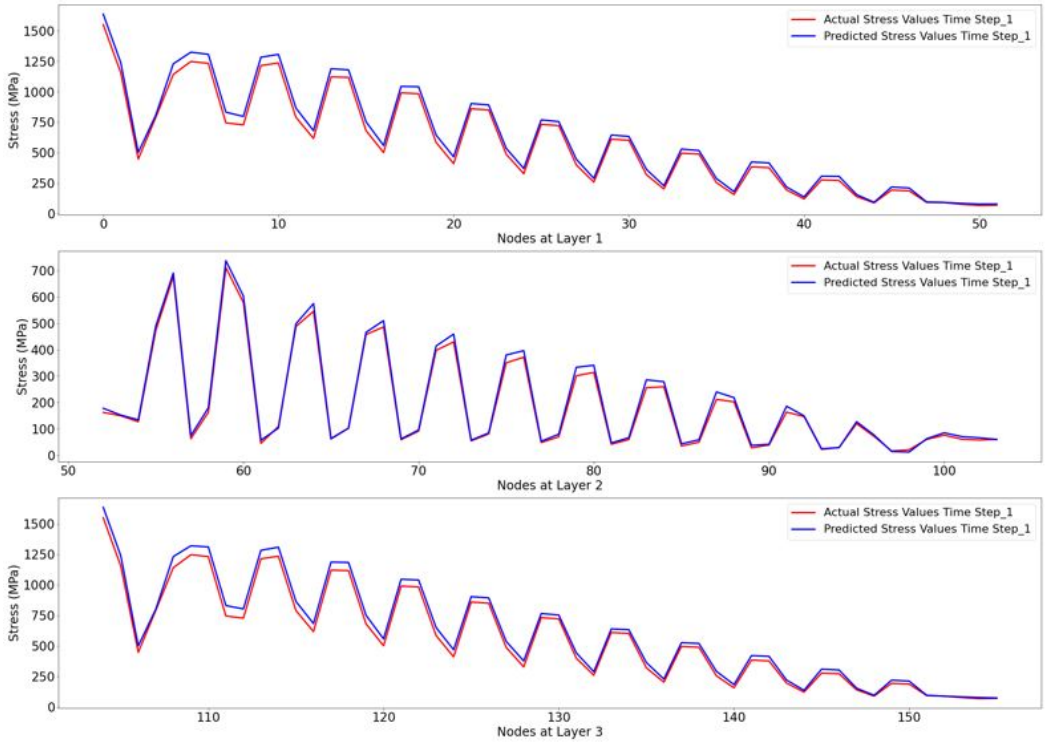

FIGURE A.36: Nodal stresses for time step 1

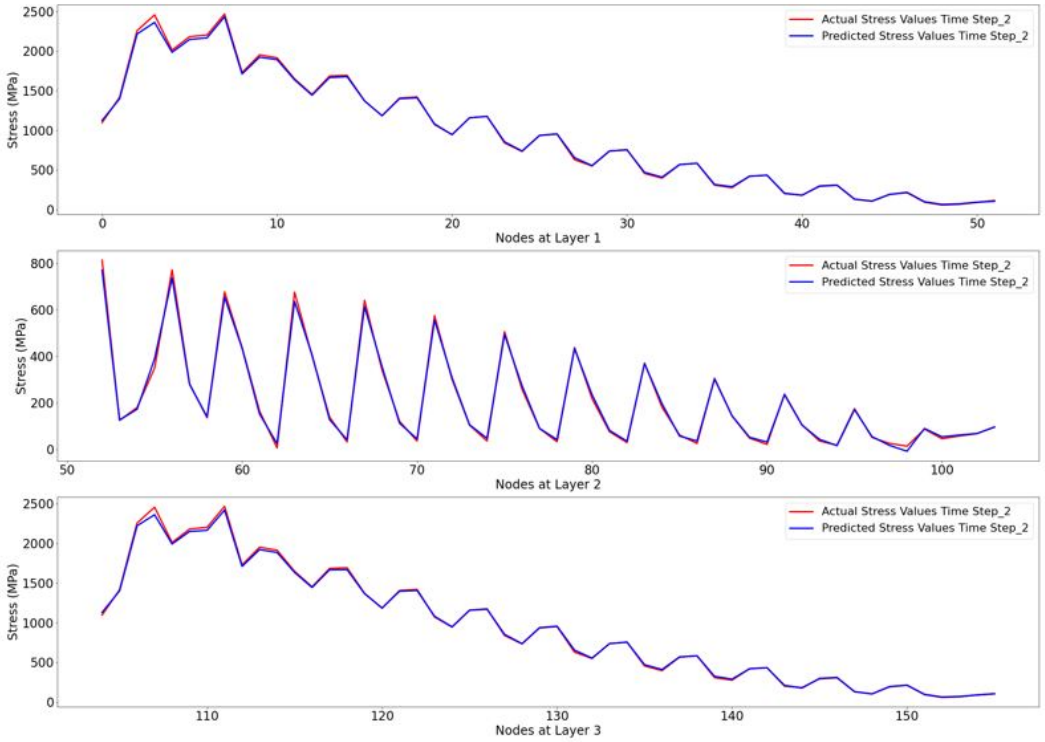

FIGURE A.37: Nodal stresses for time step 2

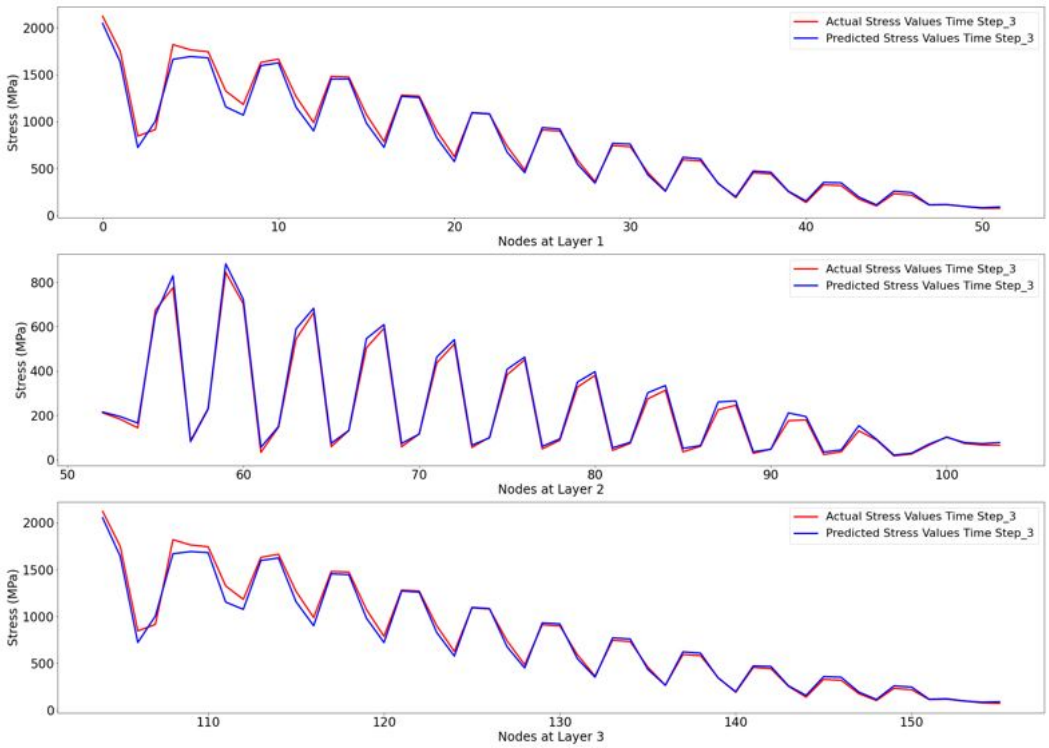

FIGURE A.38: Nodal stresses for time step 3

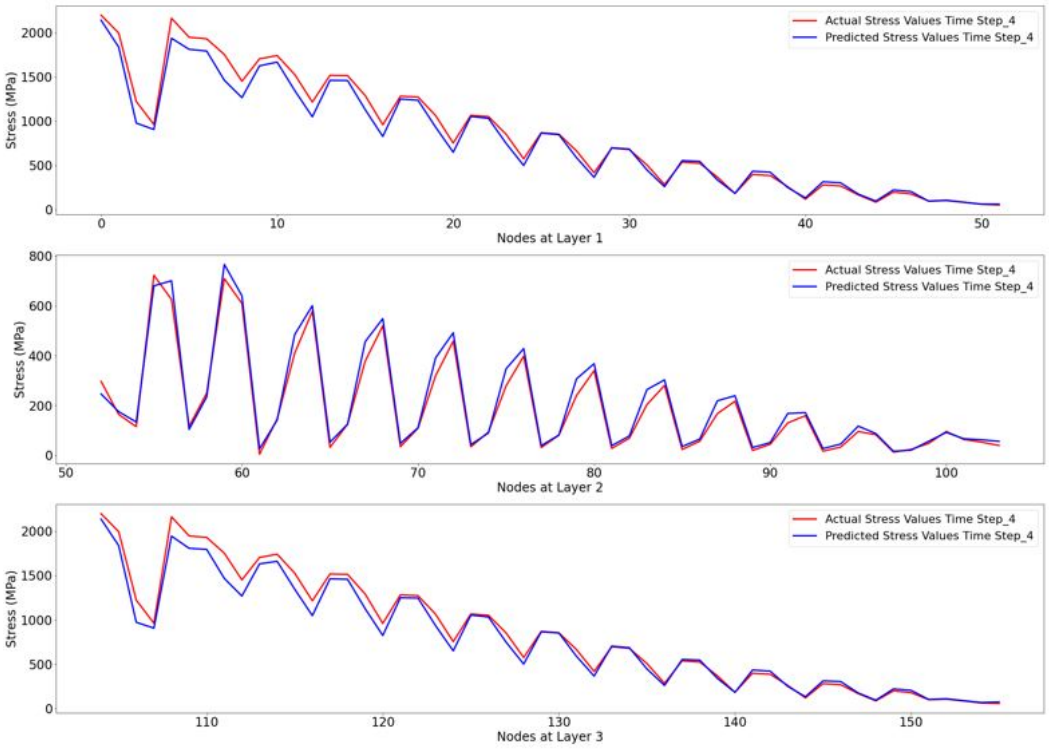

FIGURE A.39: Nodal stresses for time step 4

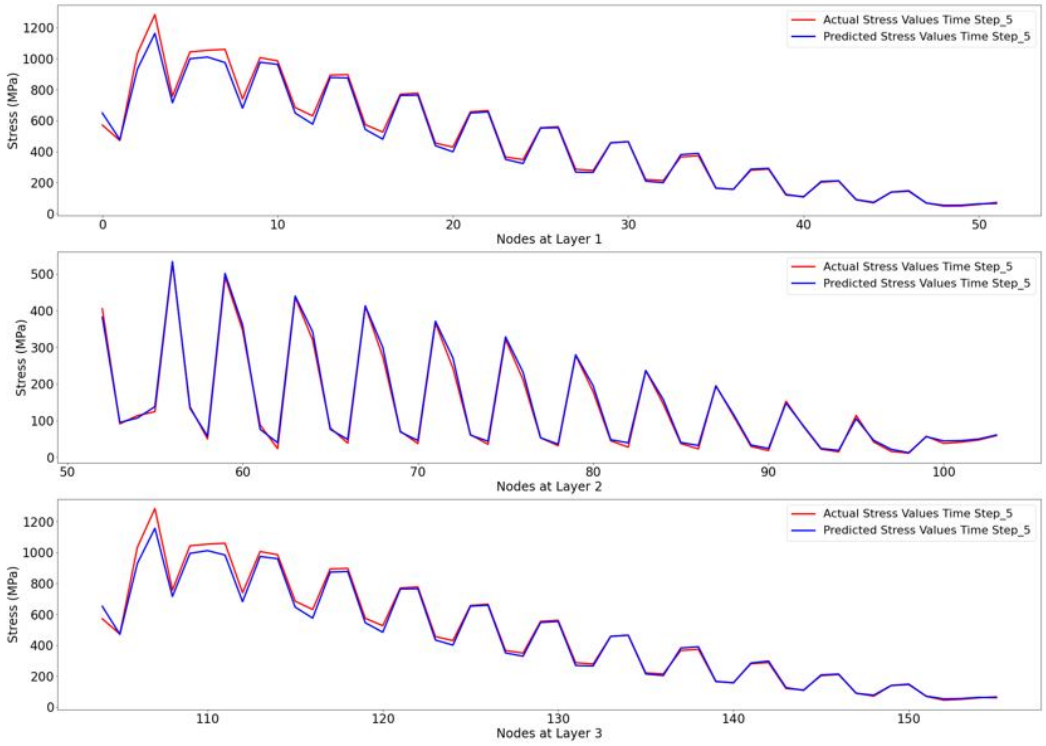

FIGURE A.40: Nodal stresses for time step 5

A minor offset in actual and predicted stress values can be observed in the plots from figure A.36 to A.40. This indicates that the performance of the GCN model in

terms of value accuracy is not the best but on a satisfactory level for this twist case.

A.5 Gated Graph Architecture on Twist Case

Plots for Nodal Z Deformation Values

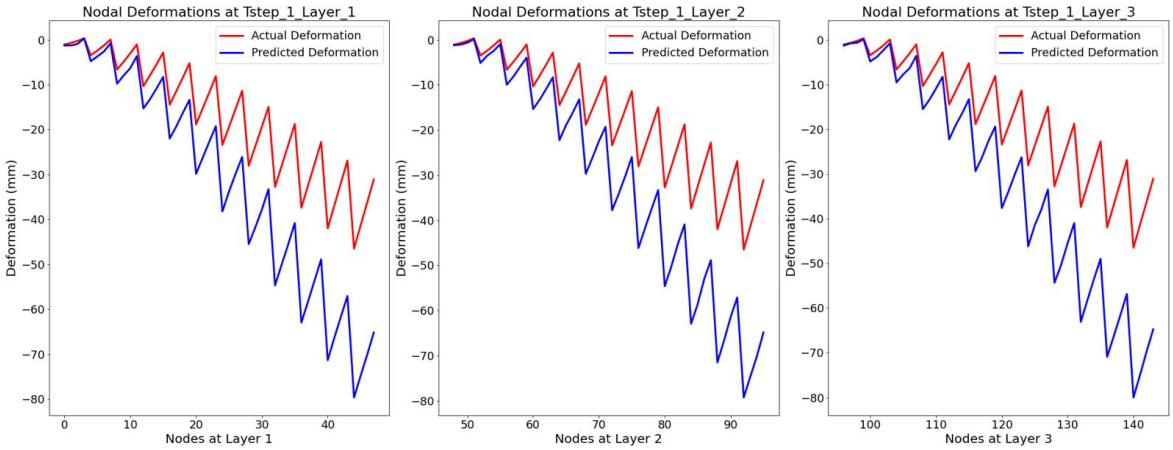

FIGURE A.41: Nodal Z deformations for time step 1

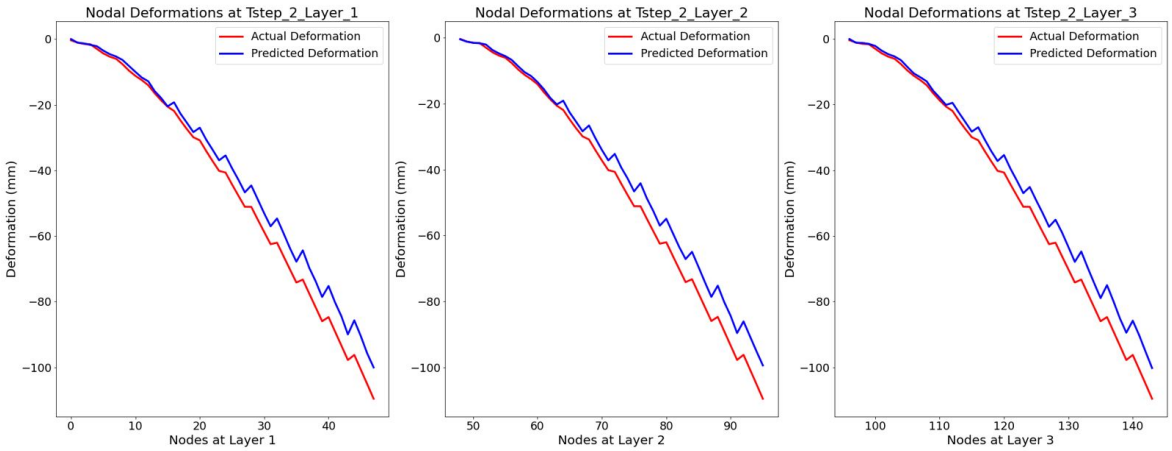

FIGURE A.42: Nodal Z deformations for time step 2

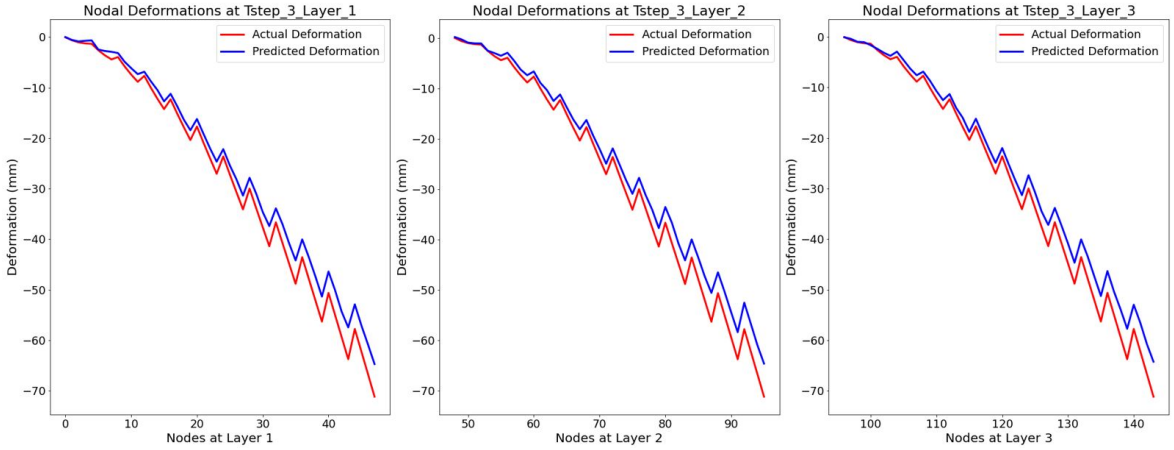

FIGURE A.43: Nodal Z deformations for time step 3

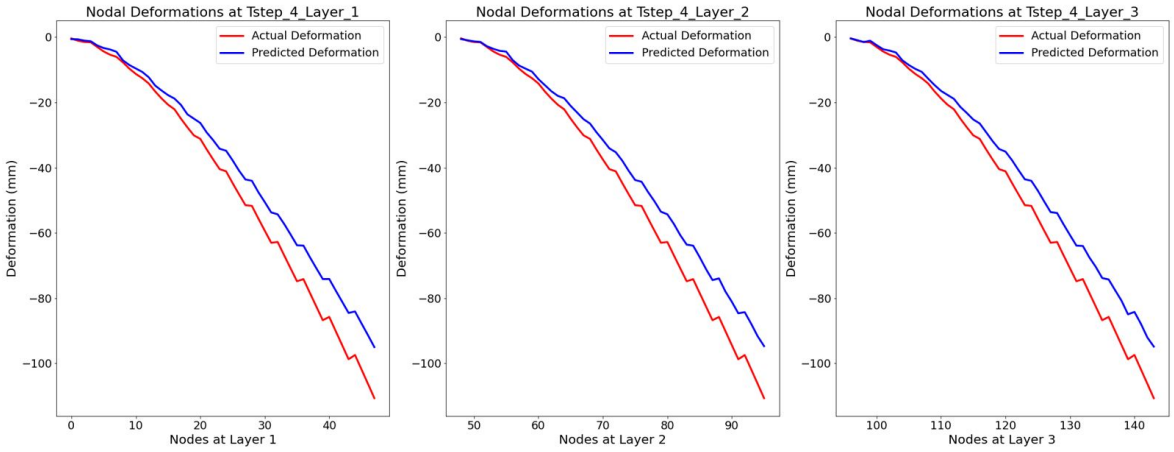

FIGURE A.44: Nodal Z deformations for time step 4

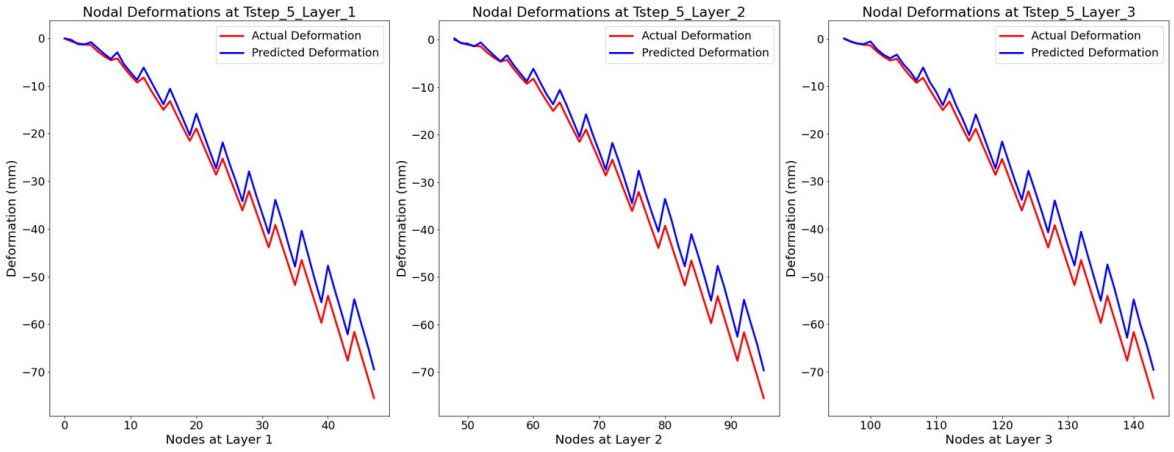

FIGURE A.45: Nodal Z deformations for time step 5

A large offset in actual and predicted values can be seen in figure A.41. Also there is no overlapping observed in plots from rest of the time steps. This indicates that the GG-NN model gives below par performance for twist case and is therefore

not suitable.

Error Visualization for X and Y deformation

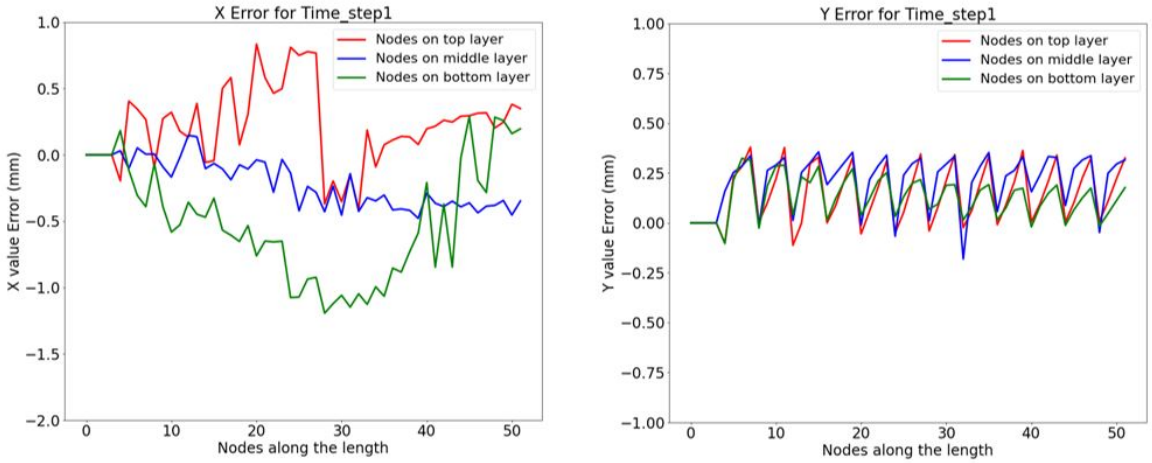

FIGURE A.46: Error for X and Y deformation at time step 1

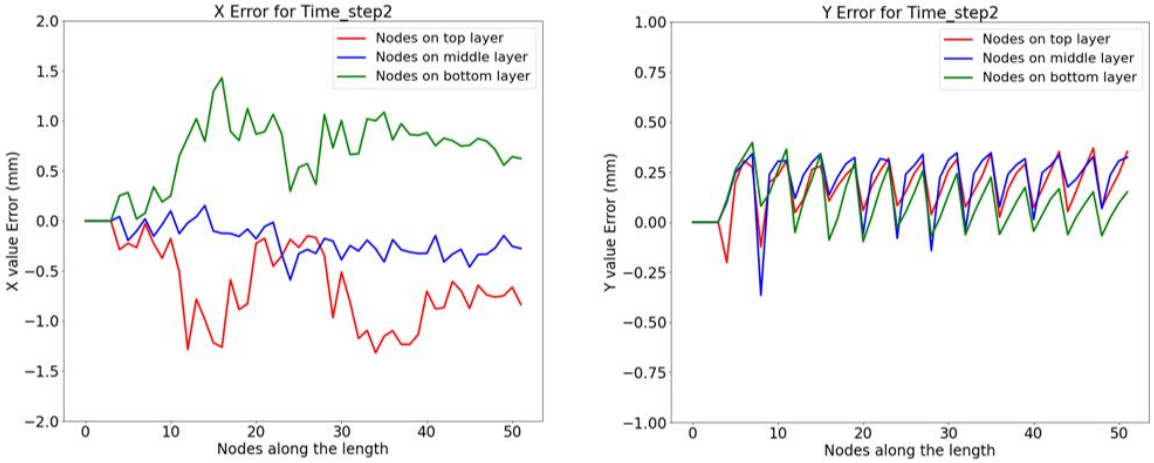

FIGURE A.47: Error for X and Y deformation at time step 2

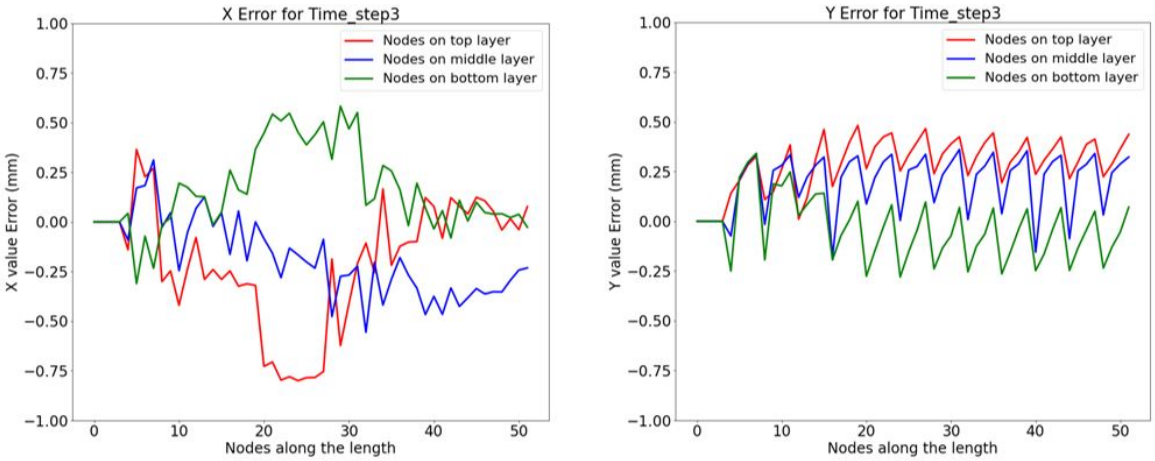

FIGURE A.48: Error for X and Y deformation at time step 3

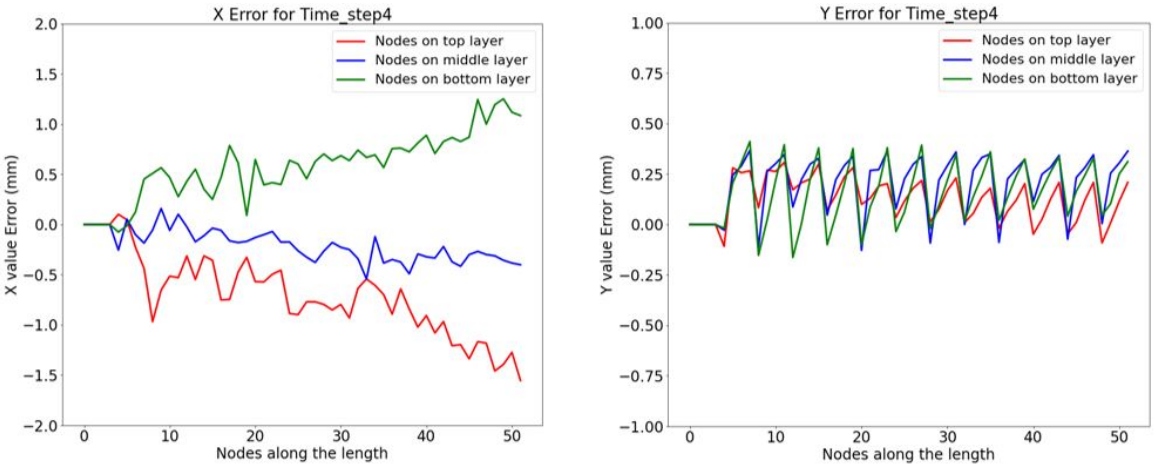

FIGURE A.49: Error for X and Y deformation at time step 4

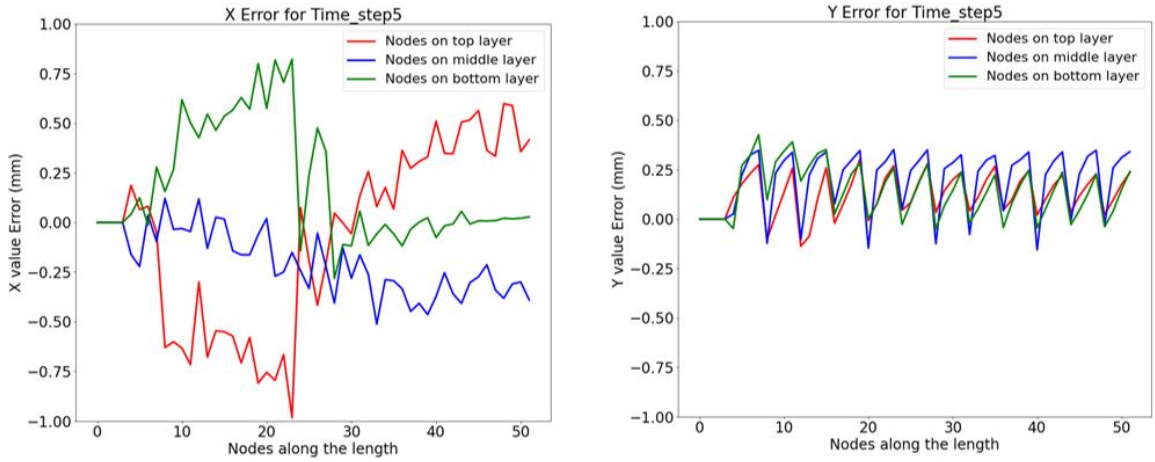

FIGURE A.50: Error for X and Y deformation at time step 5

Although the predictions for Z deformations were poor, the GG-NN model gives good predictions for X and Y values which can be inferred by observing the plots

from figure A.46 to A.50. However, for a cantilever beam, the Z deformation remains the top priority prediction.

Plots for Nodal Stress Values

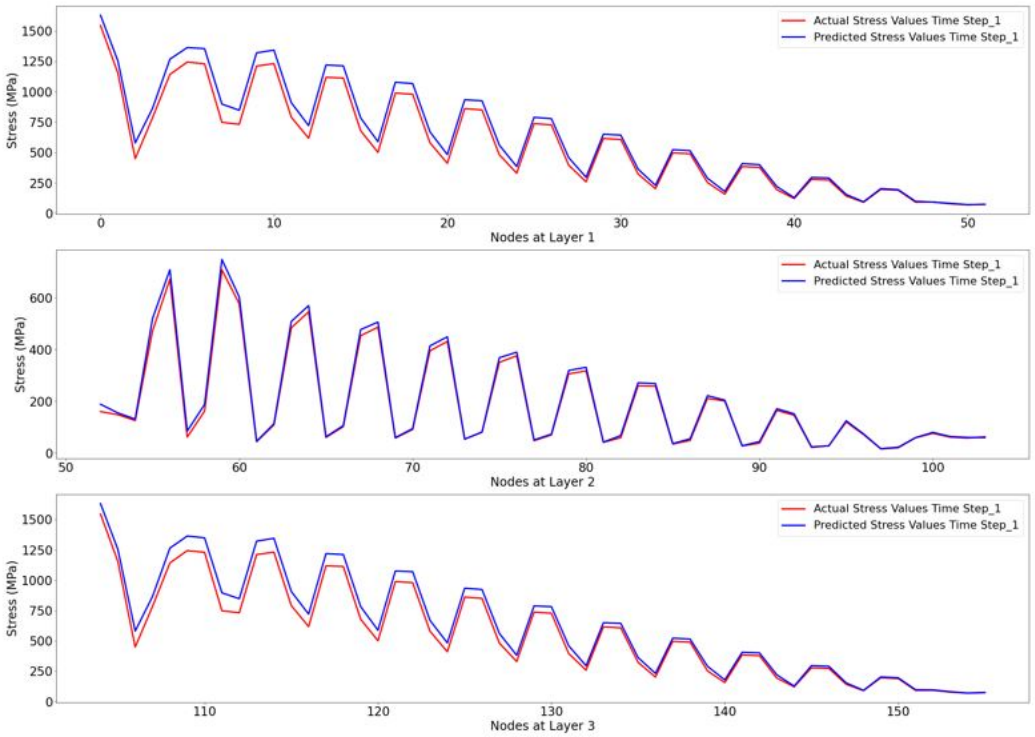

FIGURE A.51: Nodal stresses for time step 1

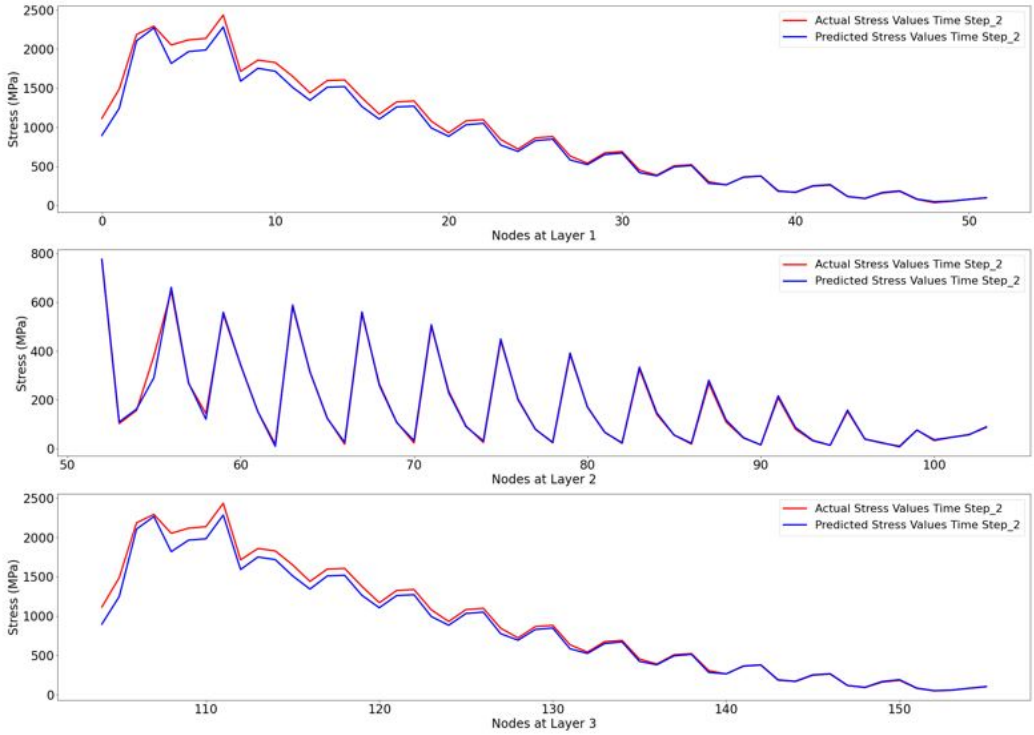

FIGURE A.52: Nodal stresses for time step 2

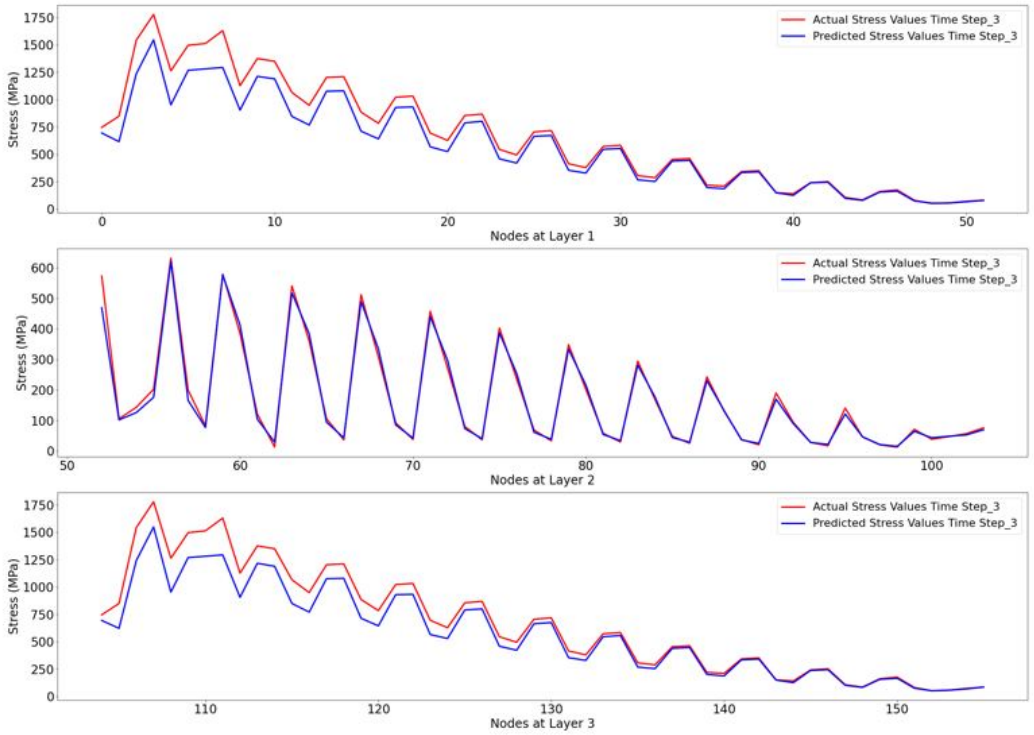

FIGURE A.53: Nodal stresses for time step 3

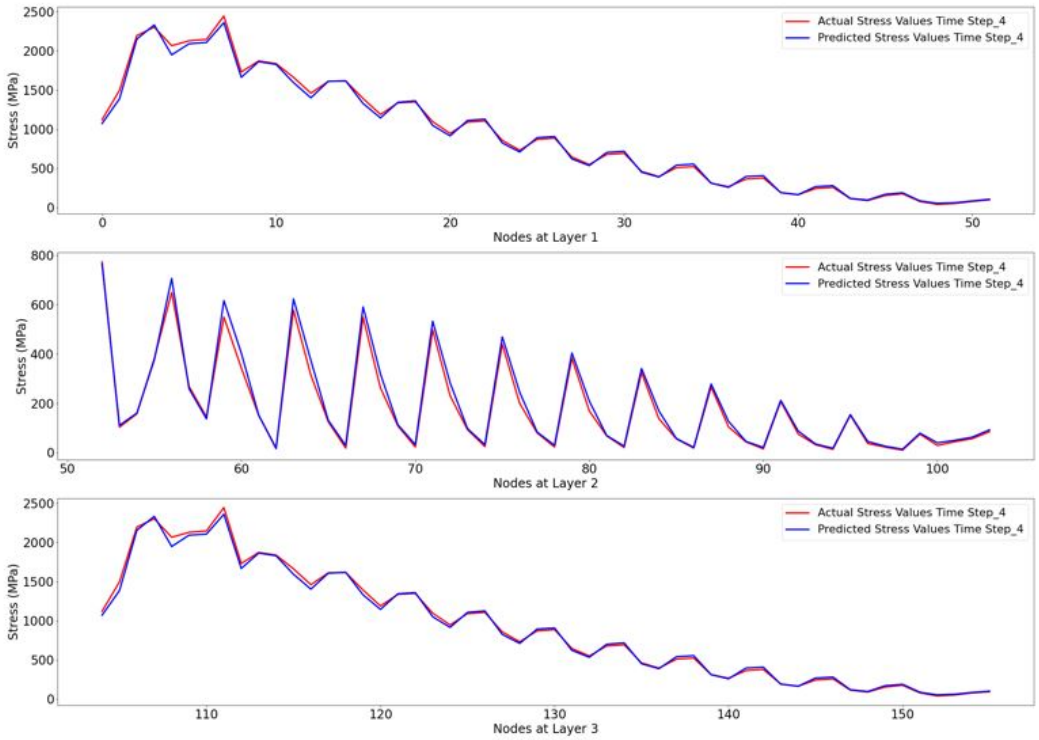

FIGURE A.54: Nodal stresses for time step 4

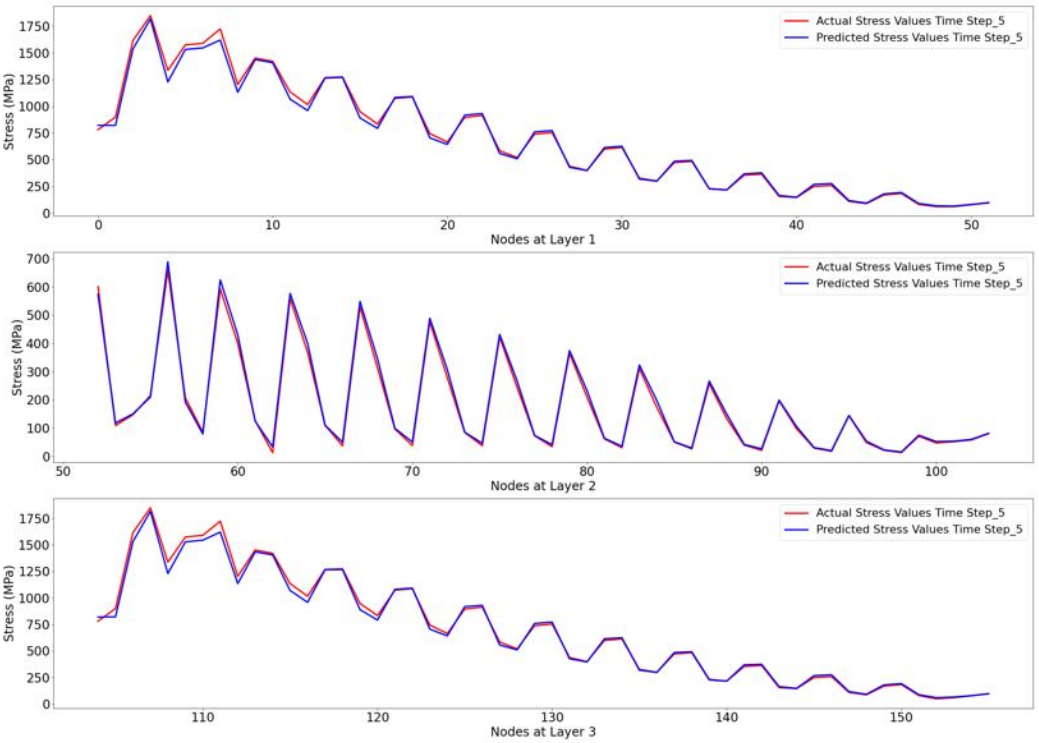

FIGURE A.55: Nodal stresses for time step 5

There is large offset observed in actual stress line and predicted stress line for time step 1, 2 and 3. This indicates below par performance of GG-NN model.

Bibliography

[1] Pacific Research, "Iterative development," 2020-09-14. [Online]. Available: https://www.pacific-research.com/iterative-product-development/

[2] GraspEngineering, "What is finite element analysis? why to do fea?" 2020-06-16. [Online]. Available: https://www.graspengineering.com/hello-world-2/

[3] K. Willems, "Deep learning in python," 2021-06-02. [Online]. Available: https://www.datacamp.com/community/tutorials/deep-learning-python

[4] I. Kandel and M. Castelli, "Transfer learning with convolutional neural networks for diabetic retinopathy image classification. a review," *Applied Sciences*, vol. 10, no. 6, p. 2021, 2020.

[5] J. Zhang, "Gradient descent based optimization algorithms for deep learning models training," 2021-05-03. [Online]. Available: https://arxiv.org/pdf/1903.03614.pdf

[6] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, no. 1, pp. 57–81, 2020.

[7] J. Leskovec, "Machine learning with graphs, stanford university lecture notes." [Online]. Available: http://snap.stanford.edu/class/cs224w-2019/slides/08-GNN.pdf

[8] E. Stein, "History of the finite element method – part i: Engineering developments," in *The History of Theoretical, Material and Computational Mechanics - Mathematics Meets Mechanics and Engineering*, ser. Lecture Notes in Applied Mathematics and Mechanics, E. Stein, Ed. Springer Berlin Heidelberg, 2014, pp. 399–442. [Online]. Available: https://doi.org/10.1007/978-3-642-39905-3_22

[9] B. Marr, "An explanation for industry 4.0," *Forbes*, 9/3/2018. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/?sh=525ceef49788

[10] R. Crahmaliuc, "Industry 4.0 – smart products and smart manufacturing," *SimScale*, 8/18/2016. [Online]. Available: https://www.simscale.com/blog/2016/08/smart-products-industry-4-0/

[11] FederalMinistry, "Industry 4.0," 2021-05-10. [Online]. Available: https://www.plattform-i40.de/PI40/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html

[12] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, P. Engel, M. Harnisch, and J. Justus, "Industry 4.0: The future of productivity and growth in manufacturing industries," *BCG Global*, 4/9/2015. [Online]. Available: https://www.bcg.com/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries

[13] T. von Tschammer, "Deep neural network in simulations," 2021-05-10. [Online]. Available: https://www.linkedin.com/pulse/deep-neural-network-simulations-thomas-von-tschammer/?trk=public_profile_article_view

[14] M. Refaat, "Simulation using ai and ml," *Industry Today*, 5/8/2020. [Online]. Available: https://industrytoday.com/how-ai-and-ml-crashed-the-simulation-party/

[15] UberCloud, "Three top trends in cae," 2021-05-10T16:22:01.000Z. [Online]. Available: https://blog.theubercloud.com/three-top-trends-in-cae

[16] News Center Microsoft Deutschland, "Was ist deep learning? definition & funktionen von dl | news center microsoft," 2020-04-09. [Online]. Available: https://news.microsoft.com/de-de/microsoft-erklaert-was-ist-deep-learning-definition-funktionen-von-dl/

[17] A. S. Bahman and F. Iannuzzo, "Computer-aided engineering simulations," in *Wide bandgap power semiconductor packaging*, ser. Woodhead Publishing series in electronic and optical materials, K. Suganuma, Ed. Oxford: Woodhead Publishing, an imprint of Elsevier, 2018, pp. 199–223.

[18] Siemens Digital Industries Software, "Computer-aided engineering (cae)," 2021-03-24T16:04:41.000Z. [Online]. Available: https://www.plm.automation.siemens.com/global/en/our-story/glossary/computer-aided-engineering-cae/13112

[19] "Computer aided engineering - an overview | sciencedirect topics," 2021-05-10T16:33:04.000Z. [Online]. Available: https://www.sciencedirect.com/topics/engineering/computer-aided-engineering

[20] "Computer aided engineering market size report, 2021-2028," 2021-05-10T16:36:28.000Z. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/computer-aided-engineering-cae-market

[21] J. Fish and T. Belytschko, *A first course in finite elements*. Chichester England and Hoboken NJ: John Wiley & Sons Ltd, 2007.

[22] SimScale, "What is fea | finite element analysis? documentation | simscale," 2021-01-20T11:00:00+00:00. [Online]. Available: https://www.simscale.com/docs/simwiki/fea-finite-element-analysis/what-is-fea-finite-element-analysis/

[23] R. Crahmaliuc, "75 years of the finite element method (fem)," *SimScale*, 11/26/2015. [Online]. Available: https://www.simscale.com/blog/2015/11/75-years-of-the-finite-element-method-fem/

[24] J. Fong, "The advantages of the finite element method," *IEEE*, 5/28/2019. [Online]. Available: https://innovationatwork.ieee.org/the-advantages-of-fem/

[25] Z. K. S.m., "Bridging the gap between machine learning and cae - the startup - medium," *The Startup*, 8/10/2020. [Online]. Available: https://medium.com/swlh/bridging-the-gap-between-machine-learning-and-cae-cf5cf4fd4017

[26] N. Dimitriou, L. Leontaris, T. Vafeiadis, D. Ioannidis, T. Wotherspoon, G. Tinker, and D. Tzovaras, "A deep learning framework for simulation and defect prediction applied in microelectronics," *Simulation Modelling Practice and Theory*, vol. 100, p. 102063, 2020. [Online]. Available: https://arxiv.org/pdf/2002.10986.pdf

[27] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks." [Online]. Available: http://arxiv.org/pdf/2002.09405v2

[28] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, "Machine learning-accelerated computational fluid dynamics," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 118, no. 21, 2021. [Online]. Available: https://www.pnas.org/content/pnas/118/21/e2101784118.full.pdf

[29] L. von Rueden, S. Mayer, R. Sifa, C. Bauckhage, and J. Garcke, "Combining machine learning and simulation to a hybrid modelling approach: Current and future directions," in *Advances in intelligent data analysis XVIII*, M. Berthold, A. J. Feelders, and G. Krempl, Eds. Cham, Switzerland: Springer, 2020, vol. 12080, pp. 548–560.

[30] S. Yoo, S. Lee, S. Kim, K. H. Hwang, J. H. Park, and N. Kang, "Integrating deep learning into cad/cae system: Generative design and evaluation of 3d conceptual wheel." [Online]. Available: https://arxiv.org/pdf/2006.02138

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

[32] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017. [Online]. Available: http://arxiv.org/pdf/1611.08097v2

[33] Amitha Manmohan Rao, *APPLICATIONS OF FINITE ELEMENTS METHOD (FEM) -AN OVERVIEW*, 2012. [Online]. Available: https://www.researchgate.net/publication/336987894_APPLICATIONS_OF_FINITE_ELEMENTS_METHOD_FEM_-AN_OVERVIEW

[34] K. Bathe, *Finite Element Procedures*, 2006. [Online]. Available: https://web.mit.edu/kjb/www/Books/FEP_2nd_Edition_4th_Printing.pdf

[35] N. Mojsilovic, "Method of finite elements." [Online]. Available: https://www.scribd.com/document/151104628/Method-of-finite-elements-by-Dr-Mojsilovic

[36] A. P. S. Selvadurai, *Partial Differential Equations in Mechanics 1: Fundamentals, Laplace's Equation, Diffusion Equation, Wave Equation*. Berlin and Heidelberg: Springer, 2000.

[37] B. Thakkar, "Finite element analysis in vb.net," *CodeProject*, 10/1/2017. [Online]. Available: https://www.codeproject.com/Articles/1207954/Finite-Element-Analysis-in-VB-Net

[38] R. A. Moreno, "Stochastic simulation and lagrangian dynamics," 2021-04-23. [Online]. Available: http://stochasticandlagrangian.blogspot.com/2011/07/

[39] B. Torstenfelt, "Finite elements - an introduction to elasticity and heat transfer applications." [Online]. Available: http://www.solid.iei.liu.se/Education/TMHL02/Book-Bars%20and%20Beams.pdf

[40] K.-J. Bathe, *Finite Element Procedures*. Klaus-Jurgen Bathe, 2006. [Online]. Available: https://web.mit.edu/kjb/www/Books/FEP_2nd_Edition_4th_Printing.pdf

[41] K. Foote, "A brief history of deep learning," 07/02/2017. [Online]. Available: https://www.dataversity.net/brief-history-deep-learning/

[42] "Neural networks," 2021-05-18T01:19:13.000Z. [Online]. Available: https://ml4a.github.io/ml4a/neural_networks/

[43] L. Deng, "Deep learning: Methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, 2014.

[44] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. New York NY USA: Cambridge University Press, 2014.

[45] "Neural network," *DeepAI*, 5/17/2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/neural-network

[46] Jaspreet, "A concise history of neural networks - towards data science," *Towards Data Science*, 8/14/2016. [Online]. Available: https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec

[47] A. Mahmoud, "Introduction to shallow machine learning," 2021-05-03T01:35:43.000Z. [Online]. Available: https://www.linkedin.com/pulse/introduction-shallow-machine-learning-ayman-mahmoud/

[48] M. S. Ansari, V. Bartos, and B. Lee, "Shallow and deep learning approaches for network intrusion alert prediction," *Procedia Computer Science*, vol. 171, pp. 644–653, 2020.

[49] J. Zhang, "Gradient descent based optimization algorithms for deep learning models training." [Online]. Available: http://arxiv.org/pdf/1903.03614v1

[50] S. Kostadinov, "Understanding backpropagation algorithm - towards data science," *Towards Data Science*, 8/8/2019. [Online]. Available: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

[51] R. Anand, "An illustrated guide to graph neural networks - dair.ai - medium," *dair.ai*, 3/30/2020. [Online]. Available: https://medium.com/dair-ai/an-illustrated-guide-to-graph-neural-networks-d5564a551783

[52] A. Menzli, "Graph neural network and some of gnn applications," 09/04/2020. [Online]. Available: https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications

[53] J. Leskovec, "Lecture notes for stanford," 2020-01-04T05:49:58.000Z. [Online]. Available: https://snap-stanford.github.io/cs224w-notes/

[54] N. S. K, "Everything you should know about cantilever beams," *The Constructor*, 10/11/2020. [Online]. Available: https://theconstructor.org/structural-engg/cantilever-beams/167474/

[55] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks." [Online]. Available: http://arxiv.org/pdf/1609.02907v4

[56] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks." [Online]. Available: http://arxiv.org/pdf/1511.05493v4

[57] "Pytorch_geometric 1.7.0 documentation," 2021-05-08T05:44:43.000Z. [Online]. Available: https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html#convolutional-layers

[58] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2009.

[59] R. Groenendijk, S. Karaoglu, T. Gevers, and T. Mensink, "Multi-loss weighting with coefficient of variations." [Online]. Available: http://arxiv.org/pdf/2009.01717v2