TÉCNICO LISBOA

Artificial Intelligence for the Analysis of Structures in Civil Engineering

Nuno Gonçalo Brás Pinhão

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Manuel Fernando Cabido Peres Lopes
Prof. Ricardo José De Figueiredo Mendes Vieira

Examination Committee

Chairperson: Prof. António Manuel Ferreira Rito da Silva
Supervisor: Prof. Manuel Fernando Cabido Peres Lopes
Member of the Committee: Dr. Plinio Moreno López

January 2021

Acknowledgments

First of all I would like to thank my parents for their caring and for providing me everything I needed to succeed in life and to reach this point at which i graduate. I would also like to thank my sister, grandparents, aunts, uncles and cousins for their support throughout all these years.

I would also like to thank my supervisors Prof. Manuel Lopes and Prof. Ricardo Vieira for always being there to help me. Their help, support, guidance and patience is why this work was finished.

Last but not least, to my closest friends and the friends I made while pursuing this degree, that helped me grow as a person and were always there for me during the good and bad times.

To each and every one of you – Thank you.

Abstract

The analysis and design of structures rely on numerical models that are computationally very expensive in the general nonlinear case. In recent years machine learning has been used to approximate functions in many domains, so there is a possibility to use it to approximate the structure responses of new designs of structures in civil engineering. In this work, we explore data driven approaches to calculate the responses of new designs of beam structures. We consider linear and nonlinear beam models, investigate different neural networks architectures, calculate relevant structure responses and behaviours, and validate it in relation to precise numerical simulations. Our results show that neural networks can approximate the behaviour of realistic beam structures to predict the bending moments, tensions, and maximum load, all this 1000x faster than the corresponding numerical simulation. This work can be used as a tool to quickly help an engineer to validate several variants of preliminary designs of new structures before committing to a long precise numerical simulation.

Keywords

Machine Learning; Structure Responses; Civil Engineering; Beam Structures; Neural Networks

Resumo

A análise e design de estruturas dependem de modelos numéricos que são muito caros computacionalmente nos casos não lineares gerais. Nos últimos anos *machine learning* tem sido usado para aproximar funções em vários domínios, portanto existe a possibilidade de usar *machine learning* para aproximar respostas de estruturas em novos designs de estruturas em engenharia civil. Nesta dissertação, exploramos abordagens baseadas em dados para calcular as respostas de novo designs de estruturas de vigas. Consideramos modelos de vigas lineares e não lineares, investigamos diferentes arquiteturas de redes neuronais, calculamos comportamentos e respostas de estruturas relevantes, e validamos em relação a simulações numéricas precisas. Os nossos resultados demonstram que redes neuronais conseguem aproximar o comportamento de estruturas de vigas realistas para prever momentos, tensões e cargas máximas, tudo isto 1000x mais rápido do que as simulações numéricas correspondentes. Este trabalho pode ser usado como uma ferramenta para ajudar um engenheiro a validar rapidamente diversas variáveis preliminares de design em novas estruturas antes de se comprometer a uma simulação numérica precisa e longa.

Palavras Chave

Machine Learning; Respostas de estruturas; Engenharia Civil; Estruturas de vigas; Redes Neuronais

Contents

List of Figures

List of Tables

Acronyms

AF	Activation Function
AI	Artificial Intelligence
ANN	Artificial Neural Networks
BSDE	Backward Stochastic Differential Equation
CNN	Convolutional Neural Network
DL	Deep Learning
FEM	Finite Element Method
GPU	Graphics Processing Unit
ML	Machine Learning
NN	Neural Networks
PDE	Partial Differential Equation
PR	Pattern Recognition
ReLu	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SHM	Structure Health Monitoring

1

Introduction

Contents

Artificial Intelligence (AI) has proven to be efficient and effective in the solution of complex engineering problems involving the definition of design parameters, the evaluation of the engineering system response and processing decision making problems, while being able to obtain more accurate solutions in less time and with less computational effort by comparison with traditional methods [1].

Nowadays AI is used in various problems of many different areas, techniques like Machine Learning (ML), Pattern Recognition (PR) and Deep Learning (DL) have gained the most attention amongst structural engineering in recent years [1].

Techniques of ML are used to train a computer to classify a sample or predict a phenomenon, using techniques such as K-Nearest Neighbours, Decision Trees, Naive Bayes or Neural Networks, ML is used in diverse areas, from economics to try to predict market scenarios, to medicine trying to predict development and evolution of diseases. PR techniques train a computer to recognize patterns in a group of data, this can be used to recognize and learn new correlations and is commonly used for speech and image recognition, having practical uses like identifying fingerprints or license plates from images.

Deep Learning is a subsection of Machine Learning that involves the use of Artificial Neural Networks (ANN). ANN can learn from almost any type of data, requiring less preprocessing, because they learn through their own errors and can go as deep as necessary. This increases their complexity but because of it, ANN can model almost any type of problem. Since the beginning of the 2000s DL has been benefiting from advances in hardware, with help of Graphics Processing Unit (GPU)s and innovative algorithms, DL can reach better levels of accuracy in less computational time than ever [2].

Neural Networks (NN) can deal with a wide range of applications in mathematics and physics such as function regression, pattern recognition or time series prediction [3] and when enough data is available, Neural Networks are extremely useful at complex nonlinear problems [4]. Compared with more traditional machine learning techniques, NNs do not require as much effort in feature selection and input processing since they learn through their own errors.

1.1 Motivation

The analysis and design of civil engineering structures span a wide range of complex phenomena, corresponding to non-linear behaviours. Those behaviours are evaluated through sophisticated numerical models, which require some computational effort. Hence, in a preliminary design of a structure simplified methods are applied (some times hand calculations), which have a limited accuracy. For example, when designing a new bridge structure, it is necessary to take into account the forces the bridge will have to endure. These forces cause the structure to suffer some sort of displacement and if that displacement is over a threshold the bridge can break into a collapsed state. That is why in the process of design, it is necessary to guarantee a level of safety of the bridge by calculating the structure responses in the

desired scenarios.

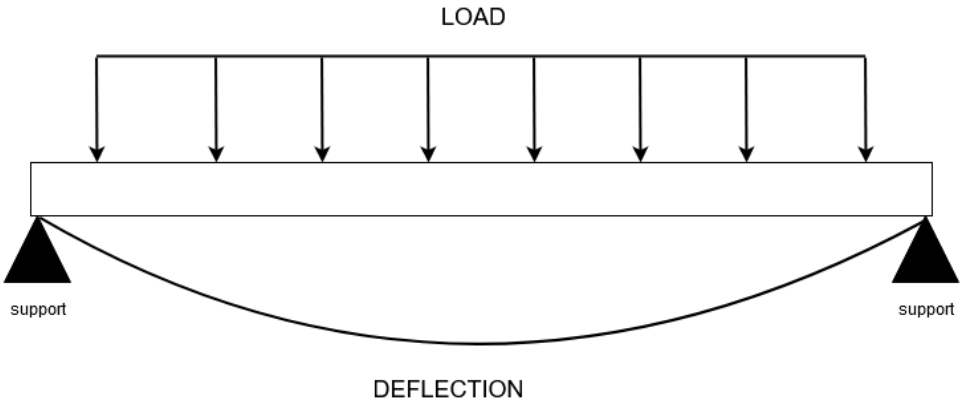

Figure 1.1: Beam diagram

However, when dealing with non linear materials or forms on the structures, complex non linear equations need to be solved. Nowadays, conventional methods of solving these equations such as the Finite Element Method (FEM), can be very time consuming and require a lot of computational effort.

The motivation of this dissertation is to find out whether neural networks can be an effective alternative for determining the structure responses in a preliminary design of a new structure. With a precise and accurate prediction of the structure responses and behaviours, we could estimate the safety of a structure scenario without having to calculate the complex non linear equations, therefore improving significantly the computational time and effort when laying out a new structure design.

1.2 Objectives

The main objective of this dissertation is to evaluate the use of Neural Networks to aid the design of civil structures, more specifically beam structures.

To explore this, we will study different types of beam designs under different types of loads and find ways to model the Riks analysis problem as a regression problem. We then need to obtain data of relevant structure responses, such as bending moments, displacements, rotations and curvatures from those beams structures and train NN models to predict them. After that we can evaluate how NNs can improve the design process of new beam structures.

1.3 Contributions

Modeled Neural Network to predict the linear elastic analysis of Beams subject to one concentrated load or a uniformly distributed load, then escalated into predicting nonlinear behaviours and more complex

scenarios. The main contribution was predicting the behaviour and the beam load capacity of continuous beams (with two and three spans) with a nonlinear material.

The aim of the analysis through the NN is to determine the structural behaviour of steel beams, considering a non-linear constitutive relation of the steel. The loads corresponding to the yielding of the beam and to the formation of each plastic hinge and the collapse load is obtained. We also explore different Network architectures and parameters, then compare them to find the most optimal setting for each type of problem. In the end, we believe we have developed models shown to have better computational time and efficiency than traditional methods like FEM.

1.4 Document Structure

This document is structured as follows:

- Chapter 2 reviews the state of the art about Neural Networks and their use inside Civil Engineering projects while referring some related works.
- Chapter 3 presents the scenarios we will be working on and the challenges and methodologies we will be using.
- Chapter 4 describes our solution, specifying the different datasets obtained and the neural networks developed for the different scenarios and challenges.
- Chapter 5 shows our models performance in different tests and evaluates the results.
- Chapter 6 concludes this document and presents some future work that can be done.

2

Related Work

Contents

2.1 Neural Networks

2.1.1 History and Biology inspiration

Artificial Neural Networks are inspired by the way the human brain works. A human brain can process enormous amounts of information using the data obtained by the human senses (touch, sight, hearing, smell and taste). This data is processed through neurons, which pass electrical signals through them.

A typical neuron has a structure that consists of a cell body, an axon where it sends messages to other neurons and the dendritic tree where it receives messages from other neurons. When an axon from one neuron contacts the dendritic tree of another neuron there is a structure called the synapse. Synapses contain vesicles of transmitter chemical, these can be different from each other, some implement positive weights while others implement negative weights. Neurons are the primary components of the nervous system, that includes the brain, and by communicating with each other through synapse, neurons form an hierarchical structure that allows to learn functions. ANNs have a similar structure, sending input signals between nodes in an hierarchical way. The human brain can learn simply because synapses can adapt, by changing its number of vesicles or the number of receptor molecules, synapses change their effects and adapt their own weights. This can make the human brain learn to perform every kind of activity such as recognizing objects, understand language, movements of the body. Much like Human neural networks, artificial neural networks also learn their functions by adapting weights.

Although, only gaining more interest in the last decade, Artificial Neural Networks were already documented as far back as the 1940s [5] and its potential was aware back then, the problem was that the technology to make it happen just simply was not there. With the latest exponential advances in technology, especially in computational hardware, the potential of neural networks has been proved and more and more models are being developed to aid in every kind of subject and area, from economics to try to predict market scenarios, to medicine trying to predict development and evolution of diseases, or to image recognition, helping identifying fingerprints or license plates.

2.1.2 Neural Networks Definition and Structure

An Artificial Neural Network is composed by at least three layers: the input layer which is the information represented in nodes that we give to our model; the output layer that is the computed output the network predicted giving the input; and the hidden layer, which can be more than one, the more hidden layers a network has the deeper the network is.

Each Layer is made of a certain number of nodes or neurons, these neurons represent values and transmit their values as signals to the neurons in the next layer that they are connected to. Neurons use an Activation Function to compute their own value using the values that were signalled to them from previous neurons, their own weight, and bias values. The connection between the Neurons can also

vary, in most cases a NN is densely connected, meaning every neuron is connected to every neuron in the adjacent layer.

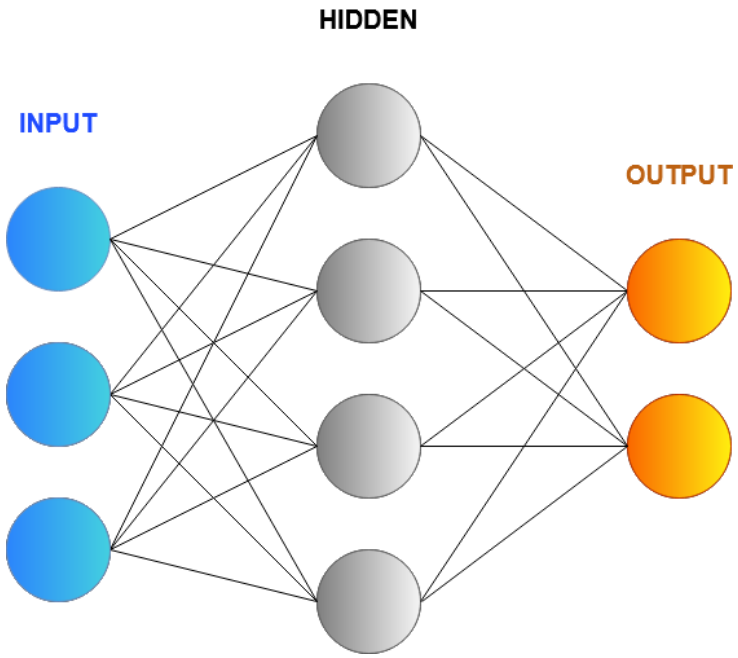

Figure 2.1: Neural Network Structure

2.1.3 Neural Networks Architecture

A Neural Network's architecture represents its number of hidden layers (depth), number of nodes or neurons in each layer (width), the connections between nodes and layers and the activation functions used in the layers to determine connection weights [6].

Choosing the optimal number of hidden layers and nodes is one of the most difficult tasks when building a NN, because too few can lead to high training error but too many can also lead to high generalization errors [2]. In [7] a correlation between a higher number of hidden layers and higher approximation accuracy was shown when solving Partial Differential Equation (PDE)s with the deep Backward Stochastic Differential Equation (BSDE) method. In [8] a network with one hidden layer composed of 30 nodes was used to predict deflection of reinforced concrete beams and in [9] a trial and error process indicated that 120 neurons for each of the 2 hidden layers used was the best architecture predicting severity indexes and locations of stiffness deficiencies.

2.1.4 Training

The process of training a Neural Network has 2 phases:

- The feedforward phase, where neurons using their weights and bias and an Activation Function compute and transmit the values to the layers ahead until the output layer computes some final value.

- Then there is the backpropagation phase where the neurons will update their values of weight and bias based on the error of the output in relation to the true value, this error is calculated by a function called the loss function, these two phases repeat until the error is low enough, each repetition is called an epoch.

Popular loss functions used in Neural Networks are mean squared error, mean absolute error or mean squared logarithmic error, the best option may vary between each problem to solve.

2.1.5 Activation Functions and Optimizers

Activation Function (AF) are used in Feedforward to, giving the weights and bias of each neuron, compute the respective output signals [2]. Across many possible AFs, Rectified Linear Unit (ReLu) has proven to be the most widely used with most success [10]. In [2] ReLu was shown outperforming Sigmoid, Tanh and Softplus activation functions, achieving a lower mean square loss in a 1000 epoch training. Compared with Sigmoid or Tanh, ReLu is more computationally efficient since it does not need to compute exponential operations. In practice ReLu tend to show better convergence than Sigmoid [11]. Sigmoid also has some advantages, specifically in not blowing up activation since Sigmoid is limited between 0 and 1. For example, in [9], a simple network with only a hidden layer and using a logsig as an activation function was used for structured health monitoring, successfully predicting the severity and location of stiffness deficiencies in buildings. However Sigmoid deal with the problem of vanishing gradient, since Sigmoid's derivative is never bigger than 0.25, when used in deep networks the N small derivates multiplied together make the gradient quickly go to zero, thus making the network impossible to train [10]. ReLu does not have with this problem since its value has no limits. So ReLu is a better activation function for networks with many hidden layers.

Furthermore, other parameters we can change to obtain the best possible prediction in less time are loss functions to evaluate our network and optimizers for Back-propagation when training our NN. Stochastic Gradient Descent (SGD) is both simplest and most popular method for solving optimization problems in Machine Learning but it requires several hyper-parameters to be efficient on some problems [2]. In [2] comparisons between optimizers SGD, AdaGrad, Adadelta, RMSProp and Adam were made, theoretically Adam should be the best overall choice and it obtained the best performance in training, however, conclusions were that the most efficient method depends on problems and conditions.

2.1.6 Convolutional Neural Networks

Neural Networks are also very popular in image recognition problems, with Convolutional Neural Network (CNN) prevailing in this topic. CNN were inspired by biological processes, their connectivity patterns between neurons resembles the organization of the animal visual cortex and making use of its pooling process and sparsely connected neurons, CNN are remarkably used for image processing in Computer Science.

Convolutional layers receive as input a high-dimensional array, that could represent anything, most commonly in image recognition problems it represents an image. Then, using 2-dimensional filters to scan the image by applying convolution operations, it returns 2-dimensional arrays called feature maps where the outputs are stored. Training a convolutional layer is simply updating the filter values since it is the filters that are multiplying on the input values. The same way we apply the convolutional layers on the input values we can also apply them on the outputs of other layers, this allows to stack multiple convolutional layers to create an hierarchical architecture that builds a convolutional neural networks.

So, making use of the distributed weights among the neurons and invariance in transitions, CNN can also be very useful to represent civil engineering structures. For instance, in our work, we tested convolutional layers that represented not an image, but a simplified structure of a beam.

Among all NN architectures CNNs have established as the most popular in civil engineering over the last years, since CNNs are capable of capturing the 2D topologies, because of a pooling process and sparsely connected neurons [1]. Recent studies have also showed that CNNs can perform better in both speed and accuracy compared to conventional AI methods and can extract and learn optimal features from raw data [1]. For instance, in [12] a CNN with only one hidden layer with 4 neurons using Unidirectional feedforward Levenberg-Marquardt back-propagation was used to correlate fatigue life with observed cracks in road bridge decks.

2.2 Artificial Intelligence in Civil Engineering

The analysis and design of structures rely on the numerical models, which are derived in order to accurately represent the real behaviour of a structure. Essentially, for the analysis of stresses and deformations of a structure, models based on the finite element method are widely and successfully adopted in structural engineering. However, in an attempt to improve some computational time efficiency, recently there has been a growing interest in applying AI in Civil Engineering projects [1]. Mainly with the use of machine learning methods, including deep neural networks, for complementing the model-based approaches, e.g. finite-element simulations, that are the state-of-the-art in civil engineering.

Today one of the biggest areas of application of AI in Civil Engineering is in aiding Structure Health Monitoring (SHM) and for the past two decades, significant progress in developing SHM models for

different kinds of structures has been made [1]. Structures can be degraded by natural hazards like strong ground shaking or wind loading, this can lead to some critical elements being damaged and some of these damaged elements cannot be visually inspected, so, the damage should be assessed by alternative methods such as non-destructive testing methods or vibration-based structural health monitoring techniques [9]. SHM involves using data collected from sensors to monitor a structure, extracting damage sensitive features, and interpreting those features for condition assessment of the structure [1]. The capabilities and the recent developments of artificial intelligence algorithms has made possible to analyse significant amounts of data measured ”in-situ” from the real behaviour of structures. Depending on the structure relevance, it is common to have the structures monitored (e.g., reading accelerations, deformations) and thus access to a valuable data representing the real behaviour of structures. The use of such data allow to have a better insight on the structural behaviour and to tune the corresponding modelling. Furthermore, due to the technological development of monitoring devices (with a significant capacity of readings) it is possible to update the numerical model taking into account real time from the structure behaviour. This model can then be a surrogate of the real structure, being updated according to the service conditions of the structure, which will allow to evaluate the future performance of the structure and will serve as a tool to determine the timing of maintenance. AI is an excellent auxiliary tool providing an efficient and faster way to deal with the structural health monitoring of structures, identifying patterns, contributing to select the most appropriate parameters to read and the corresponding location. With this information is possible to act with regard to the safety of the structure as fast as possible. On the other hand, by being capable of analysing the significant amount of data from the structure monitoring, it allows to build a reliable digital twin of the structure. Without AI it would be extremely hard to use every valuable data to monitor the structure, thus in the future, AI will be an essential tool for SHM.

Some applications of NN in SHM are, for example, in [9] where a NN is developed based on data from a scaled down model of two buildings, based on different simulations that provide the behaviour for different stiffness deficiencies it is possible to learn a neural network that predicts a severity index and their locations, results are latter verified in a scaled down model of two buildings. With a similar approach, in [13] a simulator is used to compute the damage on the structural properties of a bridge. This simulator is then used to train a neural network aiming at detecting the severity of the damage and its location. In [14] neural networks are used to learn the relation between the natural frequencies and mode shapes and the elemental stiffness parameters. In most cases these methods can then be used in real-time to provide fast diagnostic about the structure. For instance, in [12] a NN provides a quick assessment of the life assessment of the bridge using the observed crack patterns without having to run the full simulation.

2.2.1 Neural Networks at Predicting Structure Behaviour

Another line of research that has been much less explored is to use AI to improve the calculation time of design parameters. This is important because it reduces the time necessary to verify the safety constraints of a new structure design, allowing to test more different options with little effort. Neural Networks can be a good tool for this computations [4] [8]. Today, FEM are commonly used to calculate structure behaviours, however when dealing with nonlinear examples, FEM needs to solve non-linear PDE and that can lead to solving systems of nonlinear equations, which is highly computationally expensive [15]. Since FEM approach is to discretize the whole computational domain in small regions, compute some form of solution and then gather all regions to put the whole global solution together, non-linearity can be too expensive and hard to calculate and whenever we decide to change something in the initial conditions and design parameters all the calculations must restart from the beginning [16].

Using AI, it could be possible to calculate directly the structure behaviour values or integrate AI models in the FEM to speed up the more computationally expensive steps, either way the goal would be to make calculations quicker and save both human and machine time and effort when designing a new structure, making also possible to try more alternative designs to find the best possible one. For instance, in [17] they use a NN with two hidden layers, the first with six neurons and the second with seven and using the Levenberg-Marquardt training algorithm, to ascertain the structural parameters of water harvesting structures at the conceptual stage of design, leading to accurately estimating the major design parameters in quick time. In [18] they use an NN consisting of one hidden layer having four nonlinear neurons using an hyperbolic tangent transfer function, to invert the reliability function that needs to be computed using finite element simulation. It is also possible to combine these data-driven approaches with finite-element simulations. Using data from various simulations we can learn a model to predict the behaviour in the real world. Later on, by relying on the data acquired on the long-term we can update the finite-element simulations by learning parameters from the real world. For instance, in [19], after testing an ANN based on the feed forward network and having only one hidden layer with eight neurons, for the entire structural model in predicting shock-wave loaded plates deformations and another ANN for replacing viscoplastic constitutive equations, integrated into a finite element code, is demonstrated a possibility to develop an ANN within an intelligent finite element for non-linear problems in structural mechanics, leading to significant reduction of simulation time.

For the design of civil engineering structures, the corresponding structural behaviour of its structural elements (beams, slabs columns) and the structural system as whole has to be evaluated. This corresponds to determine the distribution of integral forces and stresses, displacements, deformations and accelerations. All this data will allow to evaluate the performance of the structure under service conditions and to evaluate the safety to ultimate limit states (i.,e collapse of the structure)

In a non-linear scenario, determining such responses of beams and other structural elements can

be a computational challenge, particularly due to geometrical and physical non-linear behaviours. As for example implementing the reinforced concrete elements deflections are hard to compute because of the non-linear stress-strain relationship of and steel [8]. Neural Networks can be a valuable tool to compute these calculations. Neural Networks capabilities at prediction, approximation, grouping, interpolation have always been an aid at solving engineering problems like non-destructive testing results analysis, construction processes planning or geotechnical problems [8]. A NN, trained with only experimental data needs no identification of material parameters and can lead to lower computational efforts and even replace the whole mechanical model [19]. Some examples of applications of Neural Networks to predict behaviour of structural elements can be found. The application of Neural Networks for constitutive modelling of concrete was first proposed by [20], where an improved technique of ANN approximation learned the mechanical behaviour of drained and undrained sand. In [21] a NN-based surrogate modelling approach is presented, they use NN to approximate the residual forces of the Newton-Raphson incremental-iterative formulation of the classical Euler or Timoshenko beams of the Equivalent Beam Element. The presented model is suitable for simulating geometrically nonlinear behaviour of carbon nanotubes. In [22] extensive research into the application of neural networks constitutive modelling of complex materials was made. An intelligent finite element method incorporating a back propagation neural network in finite elements analysis was built. Proving Neural networks capacities at predicting, with high accuracy, nonlinear material behaviour such as deflection and bending moments.

So Neural Networks can be an alternative at predicting responses and behaviours of structural elements. For instance, in [4] proposed Neural Networks models predicted inelastic moment of continuous composite beams when giving the elastic moments, with significant results at saving computational effort and in [23] three Neural Networks predicted deflections in steel-concrete simply supported bridges, 2 and 3 span continuous bridges, incorporating flexibility of shear connectors, shear lag effect and cracking in concrete slabs.

2.3 Solving Partial Differential Equation Directly Approach

Another possible approach for NN predictions in structural behaviour, that we did not develop in this dissertation but that is worth mentioning, is to use the Neural Network to solve directly the differential equations. Instead of using the collected data of design conditions and deformation values and train the NN to associate them directly, we can calculate the differential equation of our problem using the loss function of the NN to penalize the fitted function's deviations from the desired differential operator and boundary conditions [15]. However, in this task of calculating deformations in structures, the function we want to solve can, in more complex examples, be a non-linear high dimensional PDE and finding algorithms for solving high-dimensional PDEs has been a difficult task for a long time, mostly due to

the difficult problem known as the curse of dimensionality [7]. However, Deep Learning approaches have recently been successfully used to solve PDEs. [7] shows PDEs reformulated using backward stochastic differential equations and the gradient of the solution is approximated by NNs, very much like the approach of deep reinforcement learning where the gradient acts as the policy function. In [24], the Deep Garlekin method is presented, where a NN is used to approximate the solution of high dimensional PDEs, the NN is trained to satisfy the differential operator, initial condition and boundary conditions, the training is made on batches of random points so its mesh free and solves the problem of meshes becoming infeasible in higher dimensions. Similar to this, in [25], the Deep Ritz method is presented, for solving variational problems that arise from partial deferential equations, like the Deep Garlekin method the Deep Ritz method has potential to work in higher dimensions and fits well with the stochastic gradient descent. In [26] a Robust Polynomial Neural Network is described, the iterative algorithm works with the purpose to find a subset of variables and a model that minimizes some criteria. RPNN is able to identify both linear and non-linear polynomial models using the same approach and has positive results even when in the presence of outliers in the training data, it also obtained high predictive ability in small data sets as well as faster training than traditional NN approaches.

[27] and [28] show data driven approaches to solve and discover PDEs with Neural Networks serving as function approximators that encode underlying physical laws as prior information. However, one of the big advantages of solving directly the differential equations approach is that collecting data for training can be much easier, because we do not need to make a lot of simulations and calculations of the whole method of finding deformations to obtain big amounts of data to train the NN. For instance in [15], using the Deep Garlekin method from [24], a solution to a desired PDE is approximated via a deep NN, using for the training data random points sampled from the domain of the function, because of this sampling method with mini batches from different parts of the domain, the computational bottleneck of grid-based methods is avoided. So, previously computed training data might not even be necessary, in [16], solving PDEs with NN, they build a loss function for the NN that matches the desired PDEs to be used in training. The training is unsupervised, using a set of points inside the domain to train the NN and to validate during training.

3

Problem Presentation and Methodologies

Contents

This chapter explains the problem in more detail and presents the scenarios we considered for this work as well as the most important parameters we want to predict and the methods we used to obtain them. One of the most adopted and conventional methods for the analysis of structures is the FEM, for example ABAQUS, the software used to obtain the data used in this dissertation, is one of the most popular software to simulate civil engineering structures and it uses FEM. FEM is a numerical procedure to determine approximate solutions of differential equations by subdividing the domain into smaller parts called the finite elements and obtain approximate solutions for those smaller parts (finite elements).

Structural behaviour encompasses the solution of non-linear equations in order to accurately represent geometrical and non-linear behaviour of structures. Numerical models based on FEM can properly deal with a wide range of non-linearities, being however (and depending on the required accuracy) costly in terms of CPU time.

The goal is to verify whether the analysis of structures can be accurately performed by a Neural network in a faster way than conventional methods, namely the FEM, particularly for non-linear behaviour.

In this dissertation, the analysis is focused on the analysis of beams (assuming an elastic and an elasto-plastic behaviour) in an attempt to derive a tool for the corresponding structural analysis. That tool will have the advantage of evaluating the non-linear response of a beam in a faster way (almost instantaneously) by comparison with FEM models. This feature would make it possible to be adopted in a preliminary stage of the design.

All the data obtained to train the NNs developed in this dissertation was obtained with the use of the software suite for finite elements analysis ABAQUS.

3.1 Setting the scenarios

Before starting to test our method, it was necessary to decide on which type of scenarios would be considered for testing. So, first were considered predictions on simpler linear scenarios, because data would be easier to obtain and therefore the focus on the neural network development could be more intensive. This linear cases, done successfully, would already prove the possibility and potential of neural networks at modeling structure responses. Only after proving that initial idea, we escalated into more complex and non linear scenarios. These scenarios could already show improvement in computational time and effort of the NN model when compared to conventional FEM based procedure.

3.1.1 Parameters to be considered

A beam structure can have different structural systems, particularly a different set of support conditions, different load patterns and different cross-section geometries. Inside our beam scenarios, there are 2 main factors that can vary.

- the structural system regarding support conditions,

- the load pattern acting on the beam.

Regarding the structural system, two types were considered: statically determined and single span beams; and statically undetermined beams with two and three spans (continuous beams). So, we could vary not only the actual size in meters of a beam, but also the number of beams, that can lead to different scenarios and results.

As for the loads, besides varying the magnitude, we could have 2 main types of load, concentrated loads, corresponding to a force on one specific point of the beam, or, distributed loads along the beam, that is the same force on every point of the beam.

3.1.2 Linear Scenarios

Was then needed some form of representation to translate this information into a form capable of being read by Neural Networks. What we did was to discretize the whole beam model into X positions, where we could have a load or not, and simply place a value of 1 if there is a load, or a value of 0 if there is not any load. Figures 3.1, 3.2, 3.3 show this interpretation as well as the first three scenarios we chose for the first test, those were, single span beam with one concentrated load (3.1), 1-span beam with a uniform load(3.2) and 2-span continuous beam with one concentrated load(3.3).

There is also another big influential factor when designing a beam structure that is the type of supports. But since we only tested for pinned supports it was not relevant to try to introduce this information into the networks. Pinned supports are free to rotate but have constraints in the translation movements, while for example, a clamped support has constraints against both rotation and translation movements.

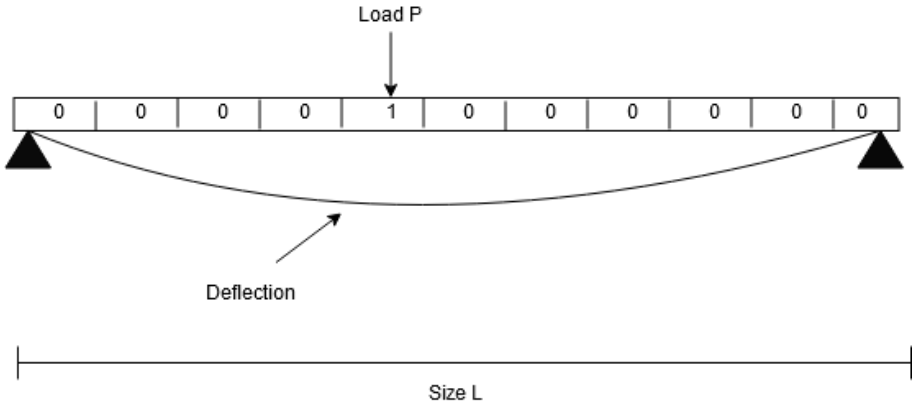

Figure 3.1: 1-Span Beam Diagram (Concentrated Load)

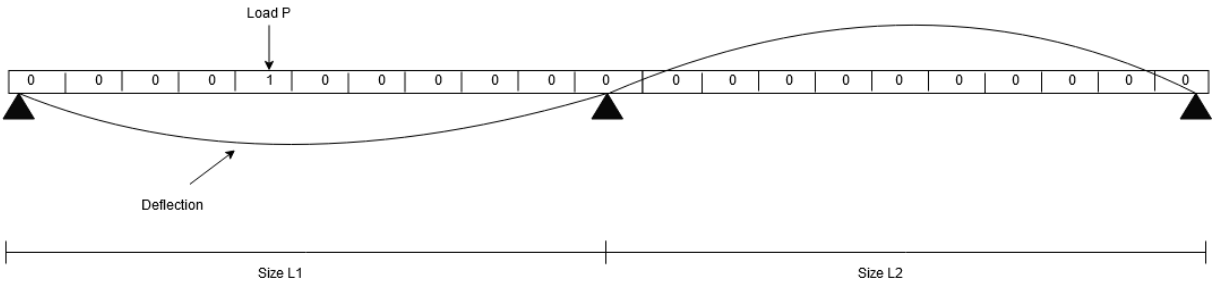

Figure 3.2: 2-Span Beam Diagram (Concentrated Load)

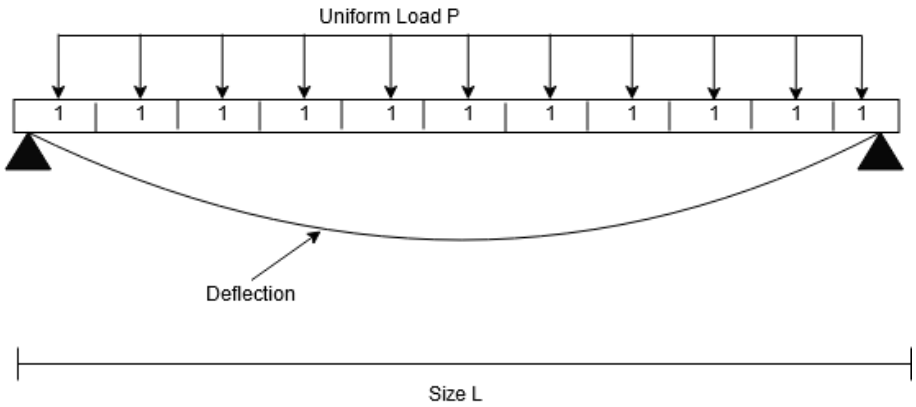

Figure 3.3: 1-Span Beam Diagram (Uniform Load)

3.1.3 Non Linear Scenarios

Further ahead, we tested our models to predict non linearity, we changed the type of material of our beams to a elastoplastic material that introduces non linearity in the behaviour and therefore in the calculations. This was the most interesting part of this dissertation, because in this example, is where we can see better the potential of neural networks computational time and efficiency, since the first linear examples are also very quickly obtained through structural analysis methods solved by hand calculations or for more complex structures using FEM, but non linear ones can take a long time to calculate.

For this example we built 2 scenarios. First a two span continuous beam with 3 concentrated forces, 2 in one span and 1 on the other spans. Figure 3.4 shows this scenario.

Then, we also tried a 3 span continuous beam scenario with one span having a uniformly distributed load and the other 2 a concentrated load in the middle. We can see an example in figure 3.5.

3.2 Structure Behaviours and Responses

Knowing which scenarios to take and which factors inside those scenarios we need to vary, now we need to know which behaviours and results we want to predict.

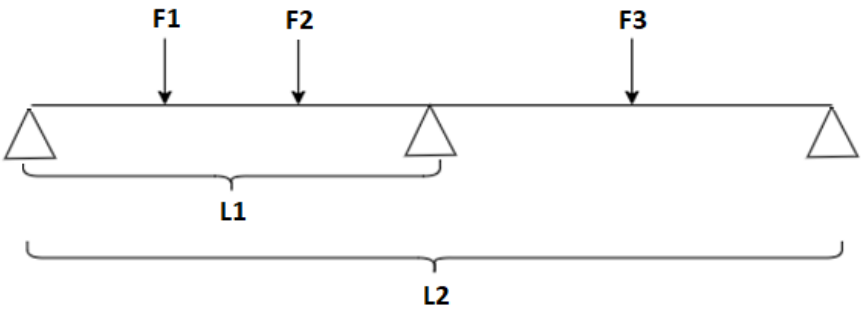

Figure 3.4: 2-Span Beam Non Linear Scenario

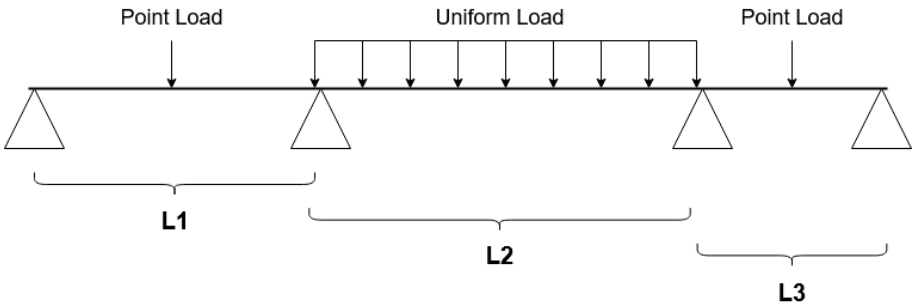

Figure 3.5: 3-Span Beam Non Linear Scenario

For the design of a beam structure (some structural systems of bridges can be modelled by a beam structure) , it is required to evaluate several parameters characterising the structural response due to the applied load (internal forces, displacements, rotations..). A wrong evaluation of those parameters can lead to designing a structure that is unsafe and that could have catastrophic consequences. As our goal is to create a model that approximate the real values of these parameters in the preliminary phase, it is important that we can accurately evaluate the most important and significant behaviours and structure responses.

The Bending Moment corresponds to an internal force associated with the beam flexure introduced by the loading. For the evaluation of beam safety to flexure a comparison between the bending moment along the beam with the beam resisting bending moments is performed. The diagram of bending moments for concentrated loads are linear segments having maxima on the position of the loads. The beam transverse displacement (deflection) shows how much the structure displaces from its undeformed shape after subjected to the load. This technically could be an important factor, because if this value is too high the structure could be on the brink of a collapse. We predicted these two responses on our linear examples, just as a proof that NNs can predict with accuracy structural elements behaviour.

Others parameters such as the Rotation and Section Curvature also allow to evaluate the structure deformation after subjected to the load. These two parameters and the bending moment will the most

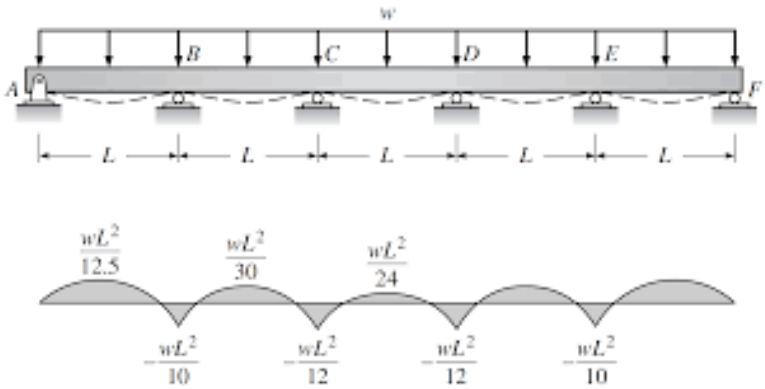

Figure 3.6: Moment and Deflection of a continuous Beam

important to predict in non linear examples.

3.2.1 Non linearity

The goal of this dissertation is to verify that Neural Networks can predict the behaviour of beam structures considering the nonlinear constitutive relations of the corresponding material, while getting results faster than traditional methods of computation. This is an interesting task because it is the nonlinear cases that lead traditional programs to their most time and effort consuming computations, so predicting this successfully and in less time could lead to significant improvements.

An important parameter to know when designing beam structure is the maximum force it can withstand safely without causing damage to its structure, which is commonly known as the beam load capacity. But nonlinear behaviour can be challenging to predict with Neural Networks. To understand why, we take the example of the material we've later chosen for the tests, which is admitted to be elasto-plastic. This non-linear material on our structure will have 3 different phases during the stress test that we want to predict. The first one is when the beam has an elastic behaviour behaviour, as load increases, some point in the beam will reach its bending yield moment and eventually will become fully yielded, this will form the first plastic hinge and we enter the second phase that is the plastic behaviour, meaning additional load will be redistributed to other parts of the structure as the hinge rotates. Eventually other hinges will be formed as the whole process repeats, until there are enough hinges to make the structure collapse (mechanism), the final third phase is after the structure collapsed, we will still have to predict this phase with NN to find out the beam load capacity.

One way to know the values of force that cause a plastic hinge is calculating results for increasing load in a scenario, and then combining the bending moment with the rotation in a plot. Because after plastic hinges occur, the plastic hinges allow free rotations to distribute the load along the beam, consequently the rotation levels increase even without increasing the load. This means we can identify the

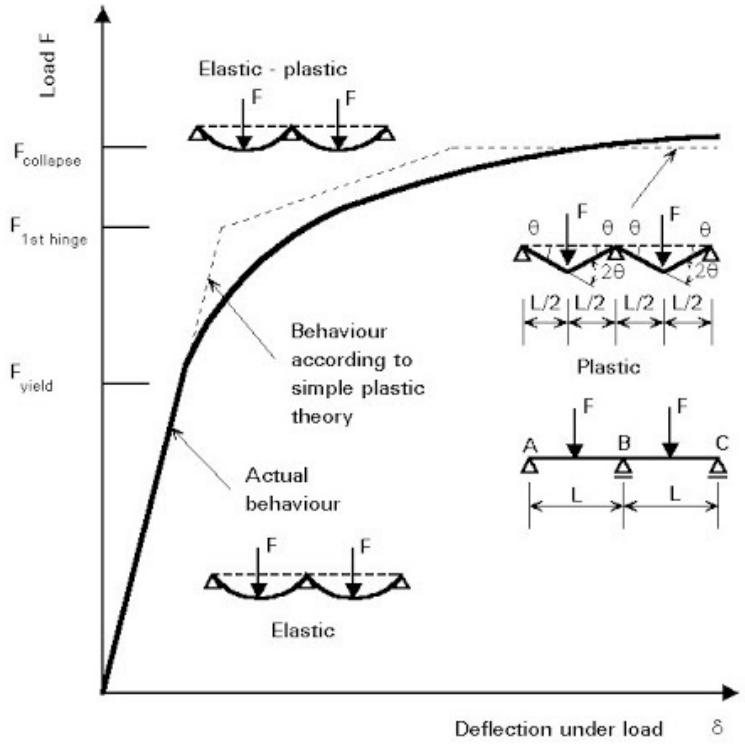

Figure 3.7: Elasto-plastic behaviour

force necessary to create the plastic hinges by plotting the bending moment and rotations together and observe when the bending moment stops increasing but the plastic hinge allows free rotations so the rotation levels keeps increasing.

3.2.2 Yield Moment and Maximum Force Supported in ABAQUS

To show it is possible to use the values of the bending moment and the rotation to evaluate the force associated with the plastic hing formation, we present an example with the data we obtained from ABAQUS.

To this end a numerical model of a beam considering a non-linear constitutive relation (an elasto-plastic behaviour) was developed in ABAQUS. The ABAQUS model considers a Newton-Raphson following path technique considering the control of the arc-length. The models output a parameter LPF, which stands for Load Proportionally Factor and represents the multiplying factor of the initial load given as an input at each arc length of the analysis. For example, if we input a load of 100kN in our analysis and at arc-length 2 the value of LPF is 4, then the actual load used to compute at that arc length is 4x100kN = 400kN. This is because the analysis computed by ABAQUS relies on an incremental procedure associated with path following strategies for the solution of non-linear equations. It keeps

incrementing the load and simulating with increasing values of load until one of 2 things happens, either the LPF reached value=9.5 (in our simulated beams LPF would never be higher than 9.5) or the load reached the maximum value of load supported by the beam. In the case it reaches the maximum load, the LPF will remain the same for the rest of the simulations, it will not increment any more.

So, in conclusion, if we give at input a big enough value of load, ABAQUS will output results for the maximum load supported by the beam and we can learn this value with the LPF. However, if we give in input a value of load that is not big enough, ABAQUS will stop at LPF=9.5 and will not simulate with the maximum load supported.

The interesting part is, that when ABAQUS reaches a value of load that causes a plastic hinge, the values of bending moment on the plastic hinge position stop increasing for the next arc lenghts until the end of simulation. But, the values of rotation keep increasing, because the plastic hinge allows free rotations (even without increasing the load, so this will also happen on the last plastic hinge). This is interesting because it allows us to plot rotation and bending moment together to find out the loads necessary to create the plastic hinges.

To show an example, for the same beam design, we did twenty simulations in ABAQUS, each with a different value of load, loads changed from 30kN to 600kN, increasing 30kN in each simulation.

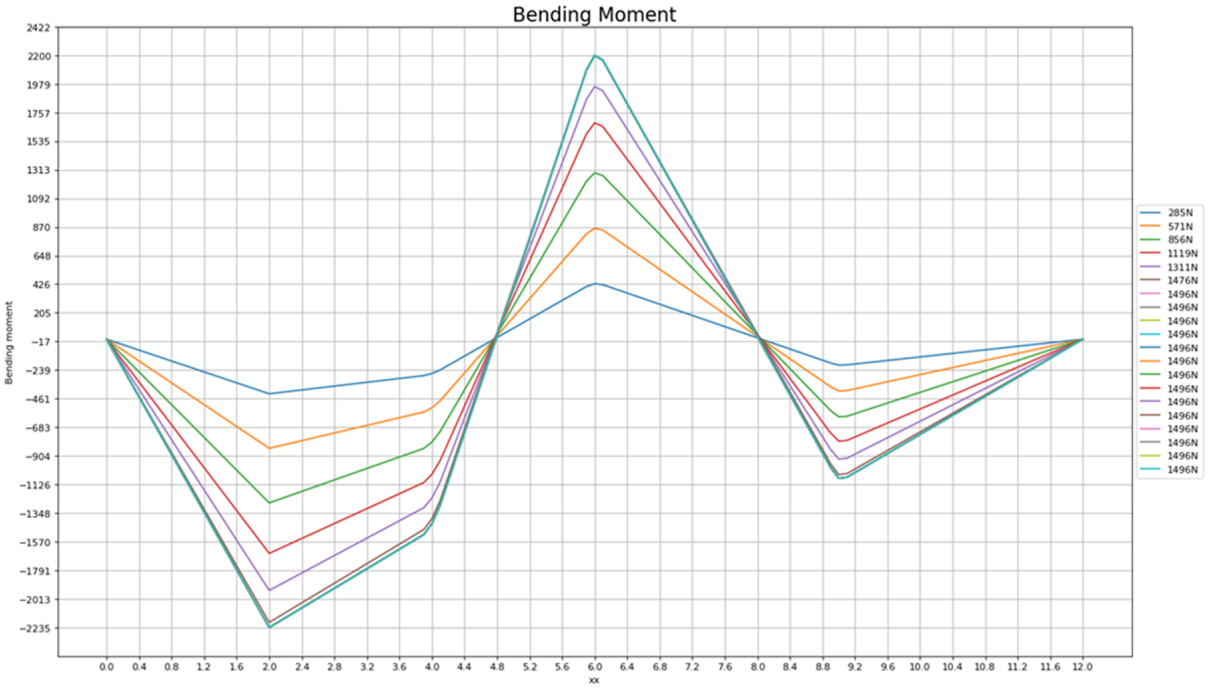

Figure 3.8: Bending Moment along the beam

The distribution of the bending moment and the rotation along the beam length is evaluated in order to determine the plastic hing position. As we can observe in Figure 3.8, the bending moment remains the same after the load reached the final maximum value. Conversely the rotation (Figure 3.9) continues

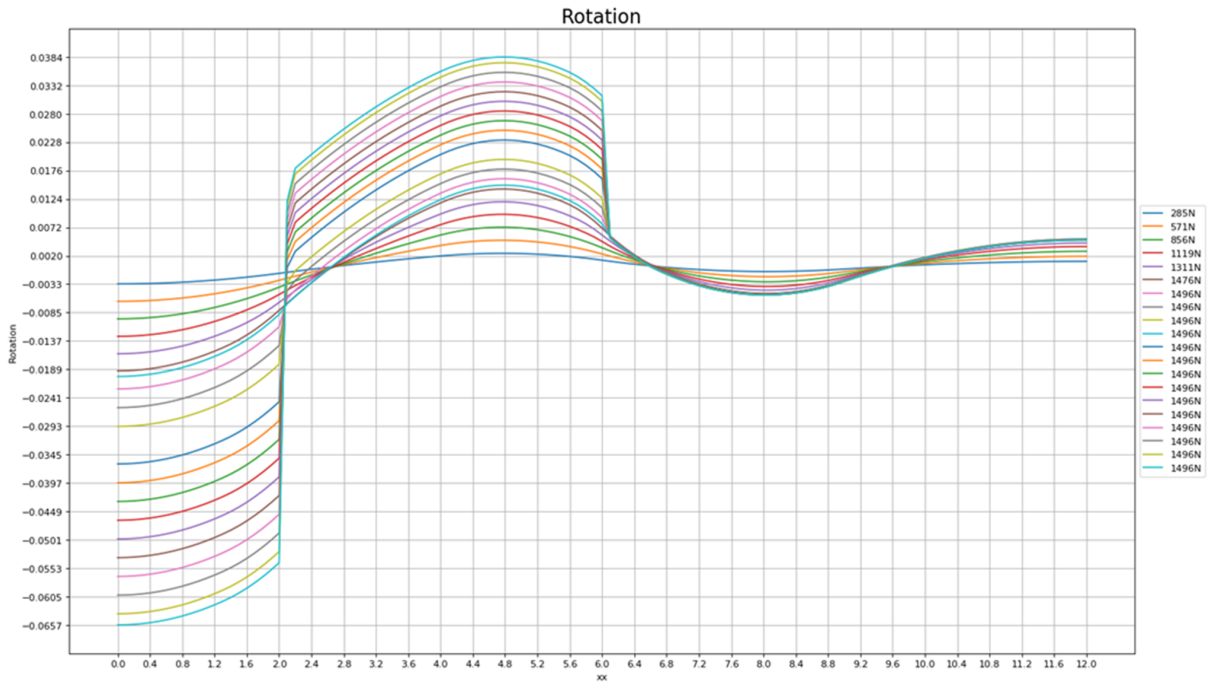

Figure 3.9: Rotation along the beam

to increase (without any increase of load), which means that a plastic hinge was formed.

This is convenient because if we know the points of plastic hinges on the beam and simulate with incrementing values of load in the input, then we can create plots of bending moment vs rotation and identify the minimum value of load to make the plastic hinges occur.

So, knowing the plastic hinges would form in position 2 and 6, we extracted the values of bending moment and rotation for those positions in each one of the 20 simulations and plotted them together.

Figures 3.10 and 3.11 show these graphs, where each point is represented with the value of load that originated it. It is simple to observe that at a certain point the slope of these graphs becomes null which is due to the fact of the bending moment stopped increasing but the rotation kept increasing. Looking back to how ABAQUS works, we can understand that this means the load has reached the maximum load before a plastic hinge occurs. Since the value of load that originated this point is 1609kN for the position 2 graph but for the position 6 graph it is 1795kN, it means that the first plastic hinges occurs in position 2 when load is 1609kN and we enter a elastoplastic regime, only when the load is 1795kN the second plastic hinge occurs in position 6 and that would cause the structure to collapse. It would be interesting to try to predict this with Neural Networks.

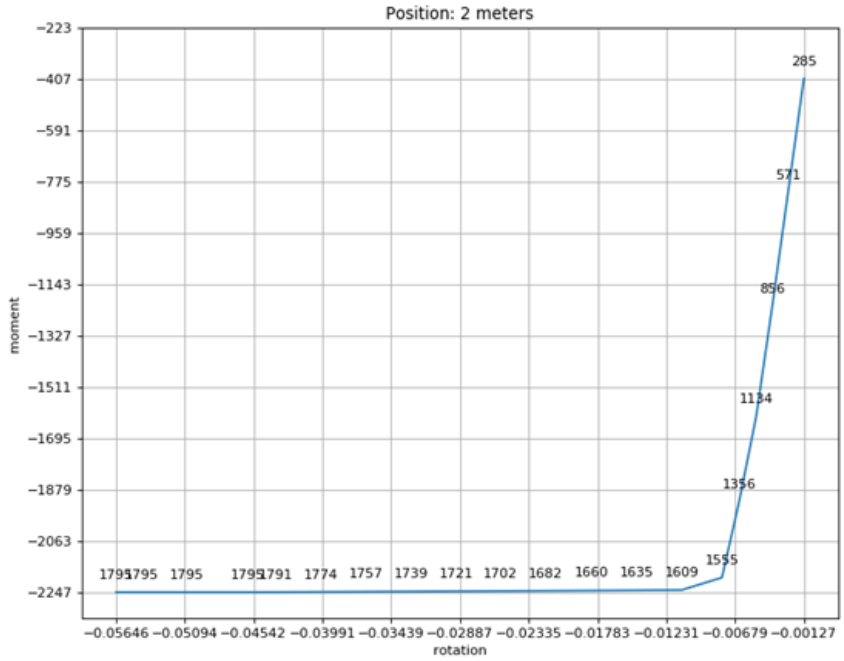

Figure 3.10: Moment vs Rotation at position 2 meters

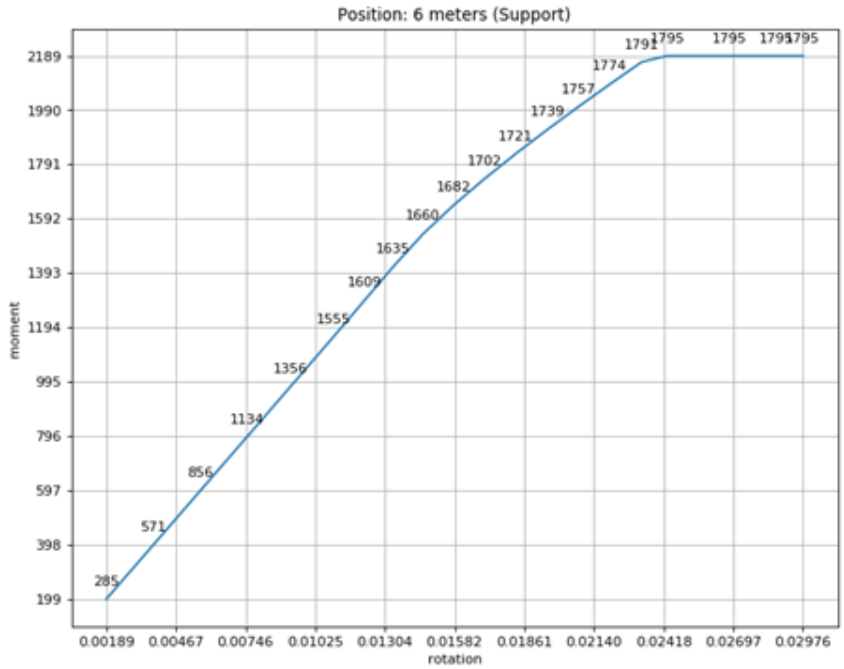

Figure 3.11: Moment vs Rotation at position 6 meters (support)

3.2.3 Using Section Curvature to identify plastic hinges positions

To this end, it would be needed to determine the positions where the plastic hinges will occur, only then we can predict the Bending Moment vs Rotation graphs on those positions and find the magnitude of

the forces that cause the plastic hinges. One can find this out by looking at the section curvature values after the analysis of a beam.

Just like Rotation, Section Curvature also represents how much a node rotates. As explained before, after plastic hinges occur and as load increases, the plastic hinges will rotate to distribute the load along the beam. Observing the nodes where these values are the highest, we can assume that those are the nodes where the plastic hinges occurred.

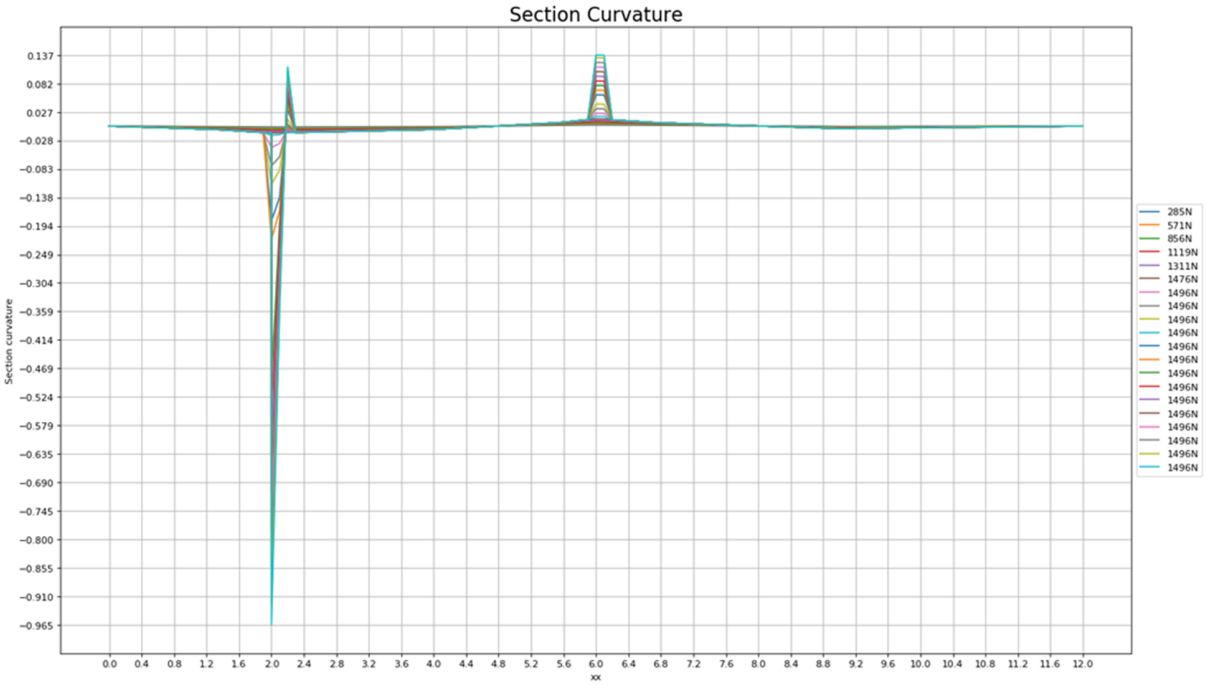

Figure 3.12: Section Curvature along the beam

Figure 3.12 is the result of twenty ABAQUS simulations of the same scenario with increasing loads. We can see that as the load input increases, the curvature value spikes in position 2 and 6, this is because in this scenario, where there were three concentrated forces with the same magnitude in position 2, 4 and 9, the plastic hinges occurred in the positions 2 and 6.

This is the method we will use to know beforehand the positions of the plastic hinges on the beam, and after that, plot the bending moment vs rotation in those positions to find out what are the minimum loads to cause the plastic hinges and consequently find the maximum load supported by the beam. All we need is Neural Network models capable of predicting accurately section curvature, bending moment and rotation levels along the beams.

4

Solution

Contents

We had settled all the scenarios and knew which metrics we needed and how to obtain them. So, now the approach resorts to simulate enough examples to create datasets for all the scenarios and metrics, then, create the neural networks optimized for each scenario and train them with the datasets, and finally use the neural networks to predict new scenarios and evaluate the results.

4.1 Datasets

In order for our networks to learn how to predict the desired structural behaviours and responses we need big enough datasets to train them. These datasets should have enough samples that represent the various examples along the domain of the problem.

4.1.1 Linear Examples

As we said before we only used the Linear examples as an initial proof that NNs can predict beams structural responses, and only predicted the Bending Moments and Deflections.

4.1.1.A 1-Span Beams

For the 1-span beams we discretized the whole beam in 11 positions equidistant from each other, the position of the load could be on any of the 11 positions of the beam except the extremity, so it could vary through 9 possible positions.

We choose the 1-span beam as our first scenario exactly because it is simple and linear, therefore to create the dataset necessary for these beams, we did not need to use ABAQUS simulations, we could simply use python to solve the polynomial equations, this helped save a lot of time in the data gathering phase.

The equations calculated to create the data necessary to train the different neural networks were the following:

A – Concentrated Load Deflection

$$D(x) = \frac{Pbx}{6EI}(L^2 - x^2 - b^2)$$

Where P is the Force of the Load, E is the Young's modulus and I is the area moment of inertia, which are all constant along the beam and multiplied by the whole formula, so we can ignore them for now because we can simply multiply them after the NN prediction. L is the size of the Beam, b represents the distant between the position of the load and the end of the beam and x represents the position where we are calculating the deflection.

The data-set included the dimension of L that varied from 1 to 101, the position of the load P that could be on any of the 9 possible positions and the resulting Deflection value in each one of the 11 positions. The data-set had 10 000 samples.

Figure 3.1 represents this scenario.

B – Concentrated Load Bending Moment

$$BM(x) = \frac{Pbx}{L}$$

This was the formula used to calculate the Beding Moment for the 1-span beam. The variables P, b x and L mean the same as in the Deflection formula. The dataset created also had 10 000 samples and varied the same variable along the same range as in the Deflection dataset.

C – Uniform Load Deflection

$$D(x) = \frac{Px}{24EI}(L^3 - 2Lx^2 + x^3)$$

And finally this was the formula used to calculate the Deflection data for the Uniform Load 1-span beam. The dataset had 10 000 samples. Variables mean the same as before and we also ignored P, E and I again for our calculations. This time, since the position of the load was constant, the dataset only included the dimensions of L, varying between 1 and 101 as well, and the values of Deflection in each position.

We can see a representation of this scenario in Figure 3.3.

4.1.1.B 2-Span Beam Concentrated Load

Then we moved forward to a more complex setting with a 2-span beam instead of 1-span, however this scenario was still linear. We only created a dataset with one concentrated load on this beam. The dataset for this scenario was composed with the size of each span (varying between 1 and 10), the position of the load, the force created by the load (varying between 1 and 100) and the resulting deflection values in each one of the 21 positions of the whole composite beam. 21 positions instead of 11 as before, because, since there is 2 spans now, each span is represented by 11 positions, however 1 position is the point where the spans meet so it is a shared position by the beams. The same representation as before is adopted, with 0s and 1 to represent the position of the load. Figure 3.2 represents this model. This time and henceforth the data was obtained with the use of ABAQUS.

4.1.2 Non Linear Examples

Since our goal is to improve the time efficiency of traditional civil engineering FEM based software, we have to escalate our scenarios into non linear problems, since it is those non linear scenarios that cost significant computational time to simulate.

4.1.2.A Scenario settings

Given the time we had, the scenarios had to be simple enough to able ourselves to create all the data necessary for the networks and build the networks that could predict behaviours with high precision. But also complex enough to be able to show some sort of generalization and potential to evolve into a more multifaceted model, that could encompass more variables and predict even more different possible scenarios.

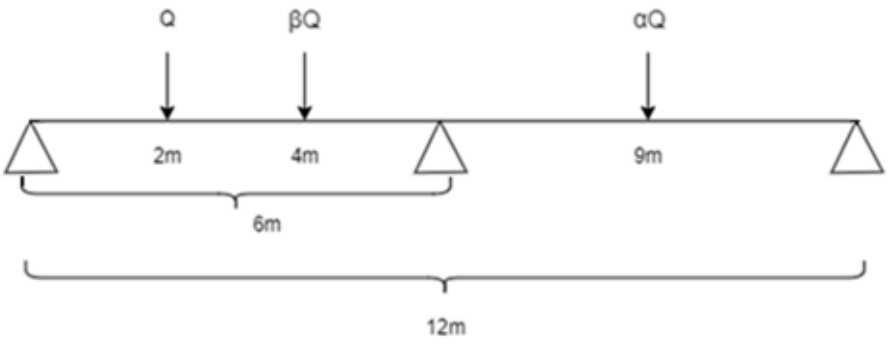

Figure 4.1: Scenario for the 2-Span Beams

A – 2-Span Beams The scenario for our first nonlinear tests is the one represented in Figure 4.1. We chose a 2-span beam for more complexity than a single beam. Each span has 6 meters, so 12 meters in total. The material chosen is elasto-plastic to introduce nonlinear behaviour. The beam is pinned on the edges, we chose this because it also adds more complexity than clamped supports. Pinned support are free to rotate but have constraints in translation movement, whether clamped supports are constraint against both rotation and translation movements. Each span has three concentrated loads, two on the first span and one on the second span, again, we added more complexity with the three concentrated loads instead of linearly distributed over the beam. Also, the first concentrated load has a force of Q while the second and third have βQ and αQ, respectively. This is where we added some room

for generalization in the model we want to build, that is, the model being able to predict various different examples, in this case the multiple options for the location and magnitude of the loads.

However, for simplicity we discretized the beam in thirteen positions, one for each meter of the beam. So, six positions for each span, with position 0 and 12 being the pinned edges and position 6 the middle support.

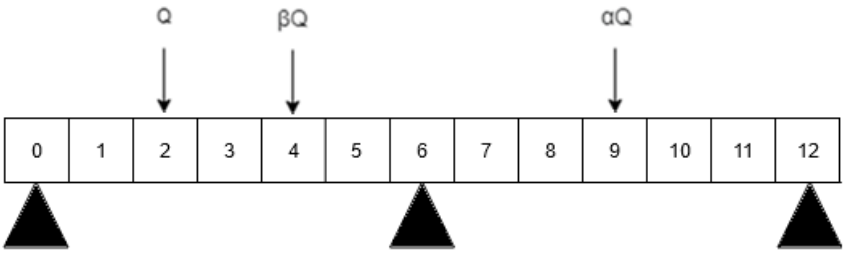

Figure 4.2: Representation of the Nonlinear 2-Span Beams

Various examples were then computed for this scenario. The parameters varied were:

- The position of the loads. The first two loads could take any position on the first span except the pinned points and the third load could take any position on the second span except, also, the pinned points.
- The magnitude of the loads. Varied from 10kN to 3000kN. β and α could take the values 0.5, 1 or 2.

We simulated in ABAQUS, 70.000 random examples inside the domain of the chosen scenario. We then, extracted all relevant input and after analysis information, including the values of Section Curvature, Bending Moment and Rotation to create Datasets for each one of these variables to predict. Specifically, we had information about the positions of the loads, their magnitudes at the input of ABAQUS and at the end of Analysis (multiplied with final LPF value) and the values of Section Curvature, Bending Moment and Rotation in each one of the thirteen nodes of the beam.

B – 3-Span Beams We then escalated into a 3-span scenario, this would be the hardest scenario we would try to predict with NN. Represented in Figure 4.3, our beams were composed by 3 spans that could vary in size from 1 to 10 meters, the material was elastoplastic as before and all the supports were also pinned. One of the spans had a uniform load while the other two had a concentrated load in the middle, the span that had the uniform load could be in the middle or on the edges, to increase even

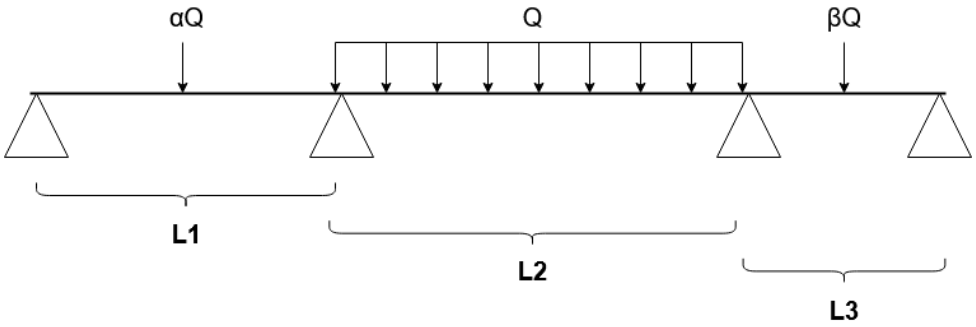

Figure 4.3: Scenario for the 3-Span Beams

more the generalization, the uniform load had a magnitude of Q while the the other two had magnitudes of α Q and β Q respectively.

To simplify and to able our NN to predict all the possible examples, we discretized the whole beam in 31 positions, 11 for each span, sharing the edges with the next one.

To create the dataset we simulated around 70.000 random examples. In each example we varied:

- Which span had the uniform load (middle or edge).
- The size of each span, varying from 1 to 10 meters.
- The magnitude of the load Q, varying from 10kN to 3700kN. β and α could take the values 0.5, 1 or 2.

All relevant data was then extracted to build the datasets to train the neural networks. Structure responses such as Rotations, Section Curvatures and Bending Moments, as well as input information such as the load types, magnitude and positions and the beam sizes.

4.1.2.B Choosing the best load type

We then used this information to train the Neural Networks developed. Training using the real load values makes sense since we want our model to predict the real value of load, and this works fine for the Bending Moment. However, knowing how ABAQUS works, for the Rotation and Section Curvature models, there will be examples that have the exact same input but different output, because of ABAQUS method of continuing to increment Rotation and Section Curvature values after reaching the maximum load supported. This will cause the NN model, given this dataset, to not be able to learn, since there are multiple different solutions for the same input. To resolve this problem we could, instead of the real load at the end of analysis, use the input load given to ABAQUS before the analysis. With these loads, the maximum load value always outputs the same value for Section Curvature and Rotation, as well as the load values should be better distributed and normalized.

However, if we also want our NNs to predict the correct maximum load supported by the beam, we cannot train with these loads, we need to use the real loads. So, we had to use the original loads when predicting the Moment vs Rotation graphs. But perhaps we can still use the input loads for the Section Curvature prediction to identify the plastic hinges locations, this was proven to not be a problem to our final model and actually the option that lead to better results, as it will be shown later.

4.1.2.C Data Scaling

Nevertheless, we still believed that the real load is not evenly distributed and normalized enough along the dominium of the dataset, so we decided that there was still some sort of improvement to be made in our final datasets. With that in mind we tested the effects of scaling our data. We tested two techniques of data normalization. The first one was using a logarithmic scale simply replacing the load value L with log(L), this ensures that the loads will be in a more confined dominium. The second technique was linear scaling simply dividing each load minus the minimum of all loads by the difference between the maximum and the minimum of all loads.

Log Scaling:

$$L' = log(L)$$

Linear Scaling:

$$L' = (L - Lmin)/(Lmax - Lmin)$$

Results can be observed in figures 4.4 and 4.5. Figure 4.4 shows the neural network training comparisons between using the original dataset and using the datasets after scaling the data with each one of the techniques. In Figure 4.5 we can observe better the difference in NN training with each one of the data scaling techniques.

Since our original dataset has much more samples with higher values of load than with lower values, meaning it is not evenly distributed across all possible values, it would be expected that log scaling would perform better since it solves this problem. However, we obtain the best results after normalizing with the linear scaling, as we can see in figure 4.5.

4.2 Neural Network Models

Having now created all the datasets for the scenarios developed, all that is left is to create the neural networks, train them with the datasets for each scenario and observe and evaluate the NN predictions.

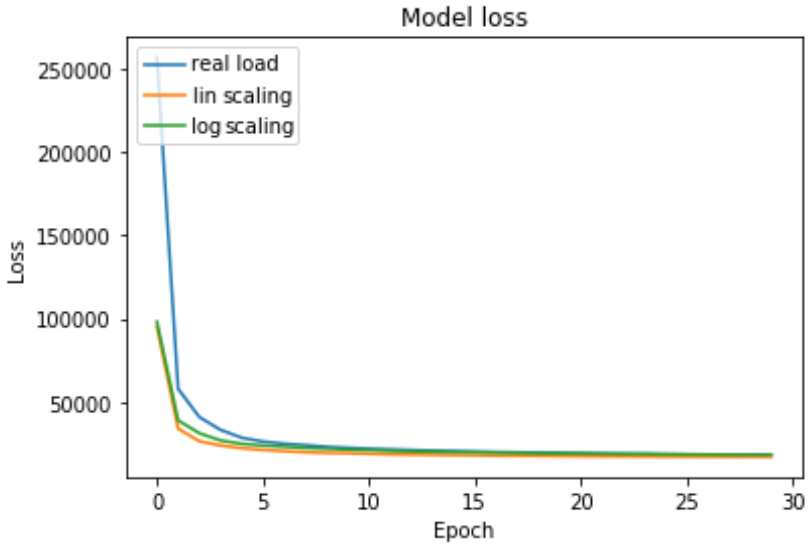

Figure 4.4: Comparisons in training between original loads before and after normalization

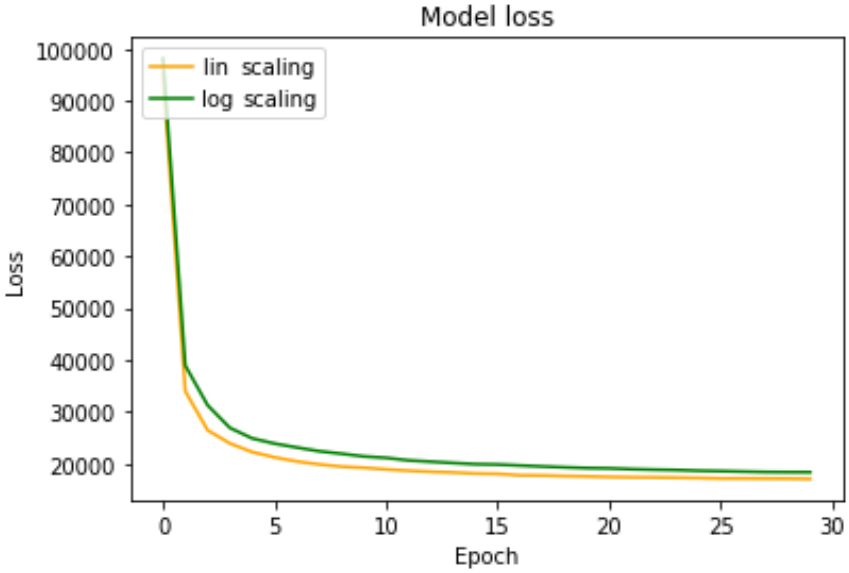

Figure 4.5: Comparison in training between the 2 normalization techniques

Neural Network models created were thought specifically to solve our problem, taking in consideration the each chosen scenario.

4.2.1 Linear examples

4.2.1.A Simple MLP Network

The first NN created for the linear examples is a simple NN, represented in Figure 4.6. This NN takes as input 12 nodes, 11 representing the 11 positions of the beam with values either 0 or 1 depending on the existence of a load and another node to represent the size of the beam. For example in figure 4.6, the 12th node has the value of L that is the size of the beam, the 7th node has value 1 while the rest have value 0, so there would be a concentrated load on the position 6 of the beam.

This Network has 3 hidden layers with 64, 32 and 24 neurons respectively.

As for the output layer, 11 nodes are computed representing the values of deflection or bending moment in each of the 11 positions of the beam.

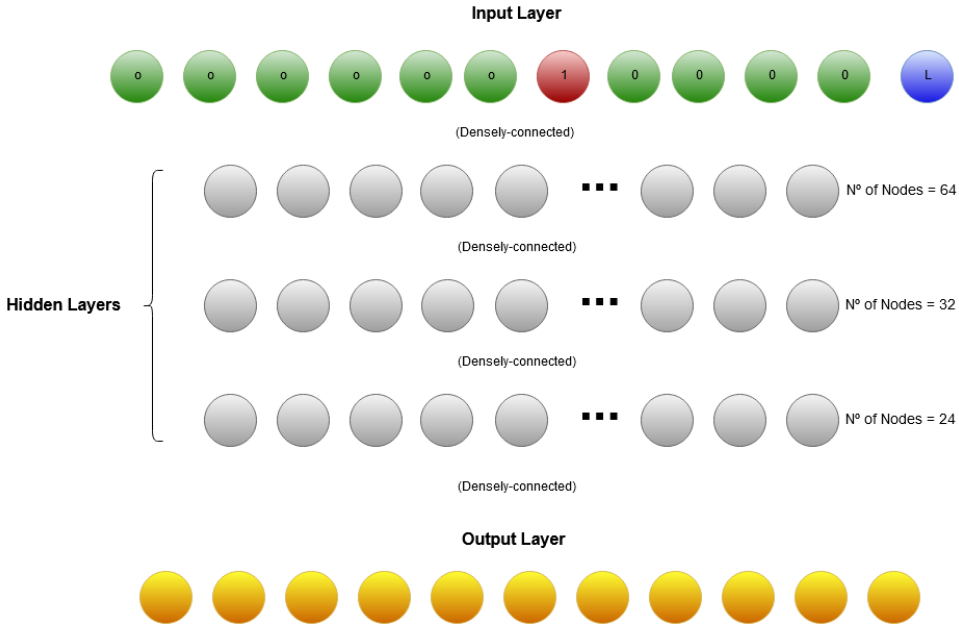

Figure 4.6: First simple NN tested for Linear examples

4.2.1.B Complex MLP/CNN Network

We then decided to test a different NN architecture, the reason for this was because since we have 11 equally distant points to represent our beam, a valid argument could be made to use CNN for its uniformly distributed weights and transition invariance.

So this network is a more complex NN, represented in Figure 4.7 and it only takes as the input layer, the 11 nodes that represent the positions of the beam, then has 3 Hidden Convolution Layers and only after we insert the node representing the size L of the beam. The next 3 hidden layers are normal densely connected layers like in the previous NN (4.6), before giving as output the 11 nodes with the

value of the structure response we want to predict (deflection or bending moment) in each one of the positions.

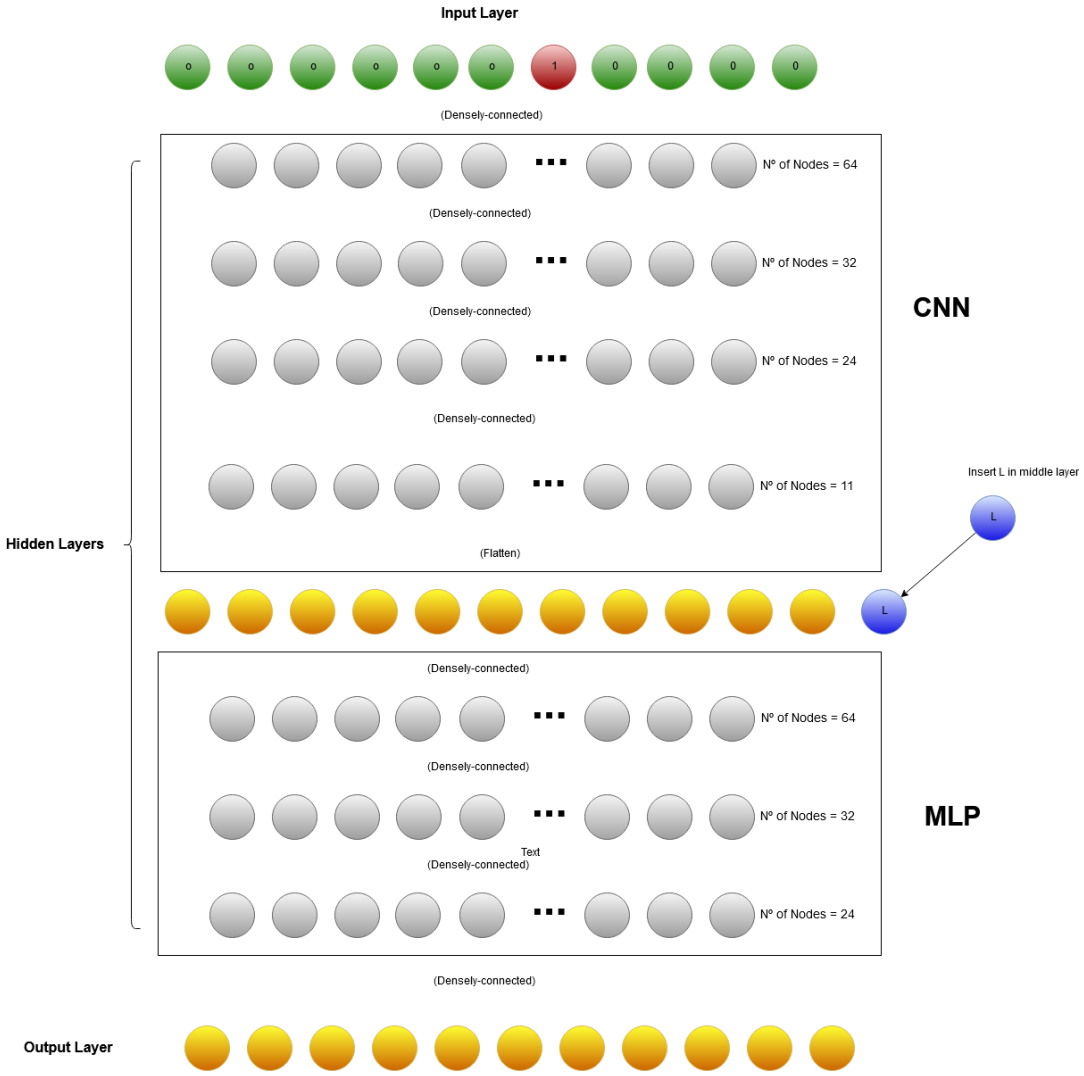

Figure 4.7: Second more complex NN tested for Linear examples

4.2.2 Non-Linear examples

4.2.2.A 2-Span Beams

From the linear examples, we believe to have found the optimal architecture for this scenario, so the first Neural Network created for the non-linear scenario (Figure 4.8) was inspired by that. It had thirteen input nodes, these nodes represented the thirteen positions of the beam and provided information about the positions of the loads, to make this, we placed a value of 1 on the nodes representing a position that had a concentrated load, and a value of 0 on nodes that represented a position without any load

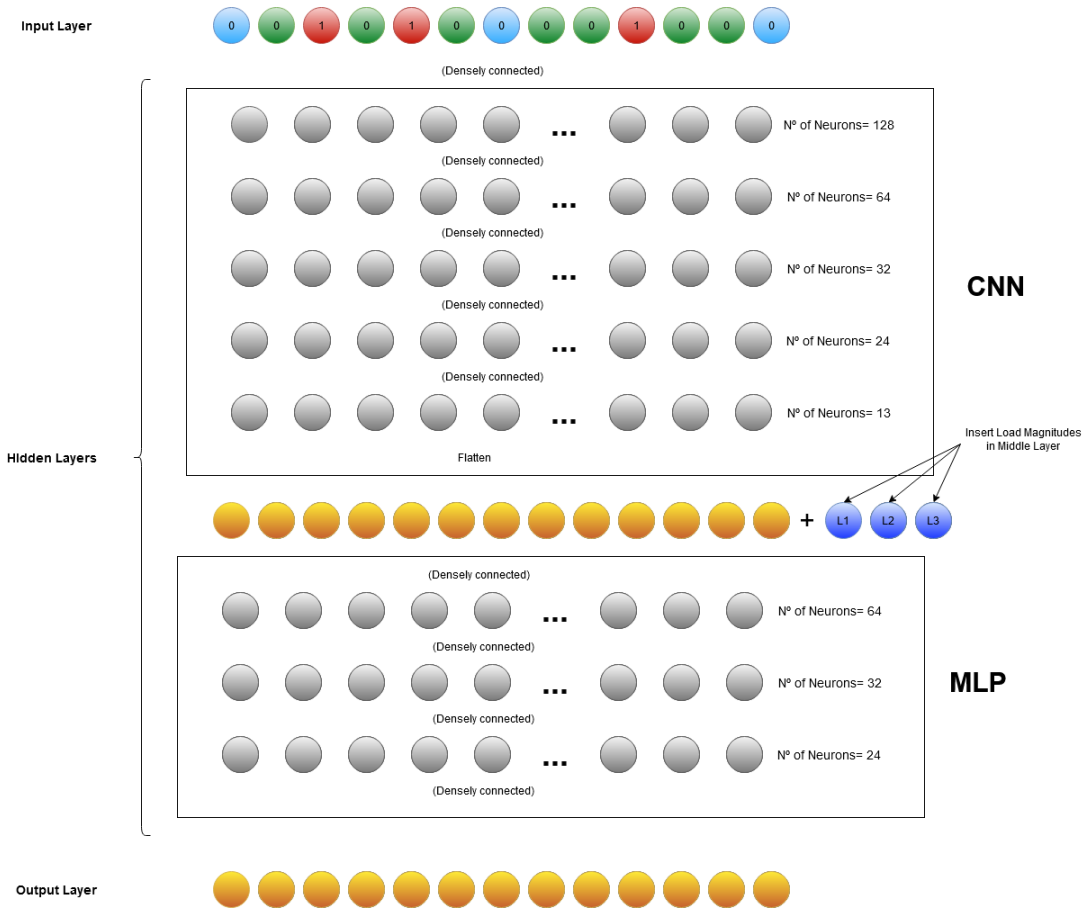

Figure 4.8: Nonlinear 2-Span Beam Neural Network Architecture

on it. This Network Layer would then be processed through five Convolutional Layers, with 128, 64, 32, 24 and 13 nodes, in that order and all densely connected before flattening. The choice to make these layers convolutional makes sense since the thirteen nodes represent equidistant positions on the beam and CNN makes the nodes have uniformly distributed weights. After that we concatenated to our so far computed output of thirteen nodes, three more nodes representing the magnitude of each load on the beam. The whole sixteen nodes were then computed by four Normal Dense Layers, of 64, 32, 24, and 13 nodes, in that order, until producing the final output of thirteen nodes, each one with the value of the variable to predict in each position of the beam. All these layers had ReLu as the activation function, that was proven to be the one with the fastest convergence. Also, the loss function with best convergence was Mean Squared Error, and the best optimizer was Adam because of the adaptive learning rate.

We used this architecture to predict the Section Curvature, Bending Moment and Rotation after an analysis of a beam inside the dominium of the selected scenario, and it was the architecture that obtained the best predictions, because it combined all the best methods to model our problem, such as the uniformly distributed weights of the convolutional layers, the ReLu activation function for faster

convergence and avoid vanishing gradient problems and Adam as optimizer for the adaptive learning rate because we have very different set of features (positions and loads) and the learning rate should be different for each.

Nevertheless, we still tried different architectures to compare with the original. Specifically, one where results were quite positive, was the architecture represented in Figure 4.9 where we would only give one input of position and obtain the output result for that given position, instead of predicting the whole beam like before. This Network had 7 input nodes, 3 representing the positions of the existing loads on the beam and other 3 representing their magnitudes, the last node would be the position we are calculating the result for. The input nodes will then get processed through five normal densely connected layers of 128, 64, 32, 24 and 12 neurons until obtaining the final neuron with the result.

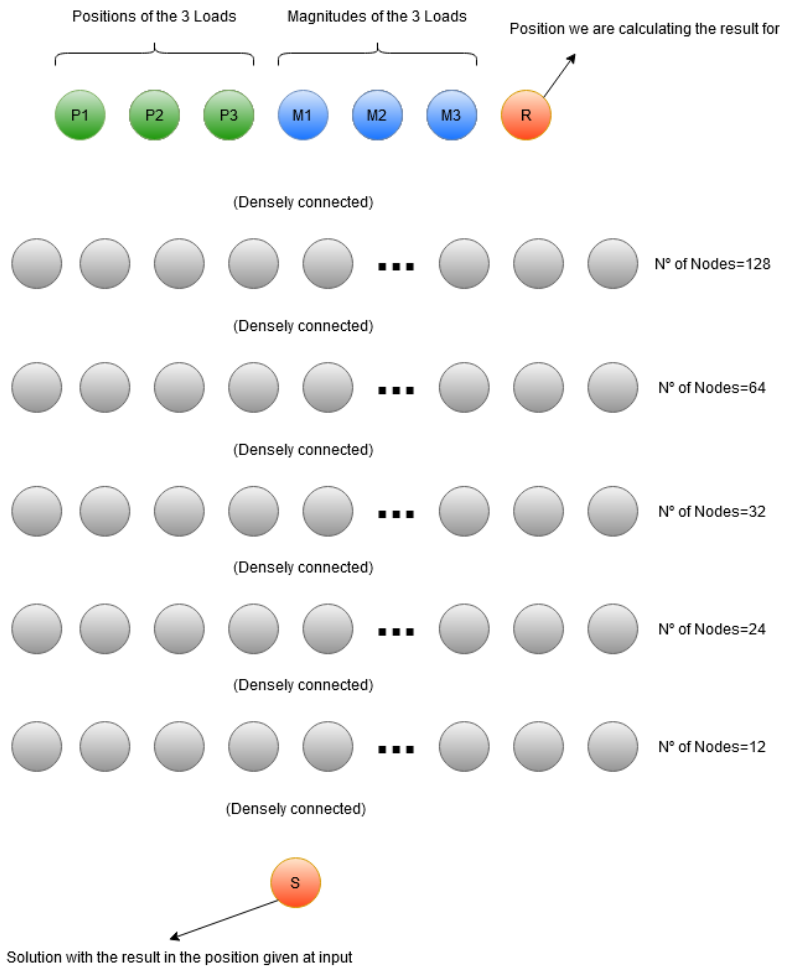

Figure 4.9: Single Output Neural Network Architecture

4.2.2.B 3-Span Beams

The NN developed for this scenario had a similar architecture as the best performing NNs in the previous scenarios with also an input layer representing the beam positions placing a value of 1 on positions that have a load and 0 otherwise, in the case of the uniform load all the positions on that beam had value 1. The input layer info will then be processed through 5 convolutional layers before adding the additional information about the size of the three beams (3 nodes) and then each load magnitude (3 nodes). The concatenated layer will then be processed through 3 normal dense layers before giving the output represented in 31 nodes, each representing a position in the whole continuous beam. Just like the NN in Figure 4.8, however the input and output layers with 31 nodes and the nodes added in the middle are 6 and not only 3.

4.2.2.C Prediction Methodology

In these non linear scenarios we went further ahead trying to predict the plastic hinges locations and the maximum load supported with the method explained in section 3.

So, to predict that, we trained 2 Neural Networks with the same architecture, one with the Section Curvature dataset and the other with the Bending Moment dataset.

Our final model takes into account a new random beam design inside the domain of the scenarios and transform the information into nodes to be input into the NN models. Then using the Section Curvature NN model it will predict the positions where the plastic hinges will occur. It finds this by giving a very high value of load in the input and the model predicts where the structure would break, therefore that is where the plastic hinges occurred. Since all we had to do was to give a high value of load, knowing exactly the value of it does not matter, and that is why we could train this model with the ABAQUS input loads instead of the real loads.

After knowing the plastic hinges positions, the program will use the Bending Moment model to predict the yield moment and the forces necessary to make the structure collapse on the plastic hinges. For that, it must predict the bending moment vs rotation graphs, so it calls the NN model multiple times to predict the same beam scenario but with incremented loads and plot the bending moment until the slope of the graph turns null, then the load that originated the point where the slope turns null is the maximum value of load. Since what makes possible to identify the slope is the stabilization of the bending moment levels, it is not even necessary to predict the rotation values and that is why we only needed the Bending Moment model.

5

Results and Evaluation

Contents

In this chapter we show every test we made to evaluate all our models in each scenario.

We started with the linear and simpler scenarios, searching for the most optimal architecture for our problems, we compared not only different numbers of layers and nodes but also different AFs, optimizers, and the use convolutional layers. We then test the results of prediction in each linear scenario.

Moving forward to the non linear scenarios, we test predictions on some of the most important structure responses and compare the 2 types of loads in the training data to understand why we have to use one type or the other on some structure responses. We also compare our NN architecture to a different one to show the increased performance. Finally we show predictions on finding the maximum load supported, we explain our model's methods showing its limitations but also its significantly increased performance compared to conventional methods.

5.1 Linear Scenarios

One of the first test we did was comparing activation functions ReLu and sigmoid for the training (Figure 5.1) and the results were similar for all the scenarios.

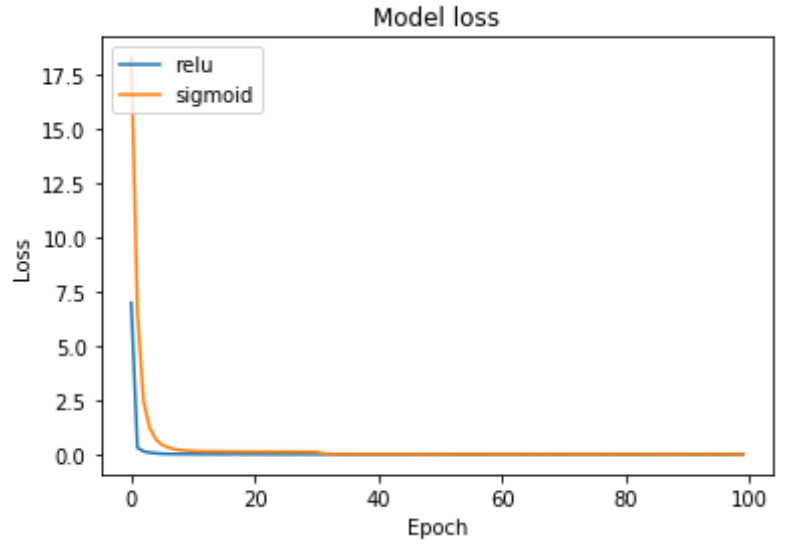

Figure 5.1: Comparison between ReLu and Sigmoid in training

It is observable that ReLu is a more optimal activation function for our NN architectures in this problem, this makes sense since we have multiple hidden layers in our networks, [10] and [2] also showed ReLu outperforming other Activation Functions. Another argument for using ReLu in our Networks is to introduce nonlinearity in the convolution layers since some of our NNs developed here use convolution layers.

So we opted to choose ReLu as the activation function for all our final tests, with MSE as the loss

function and optimizer Adam.

We had developed 2 different NN architectures for the linear tests, one with only normal dense Multilayer Perceptron (Figure 4.6) and another incorporating convolutional layers as well (Figure 4.7). Here we compare these architectures, to find if the convolutional layers make a significant difference to predict our problem.

5.1.1 1-Span Beam Concentrated Load

For this data-set, results were successful with both Neural Networks. As we can see in Figure 5.2, the complex NN with the convolutional layers (Figure 4.7) performed better, achieving the lowest error rate faster. In figure 5.3 we can see an example of a Deflection prediction in this scenario that is quite accurate.

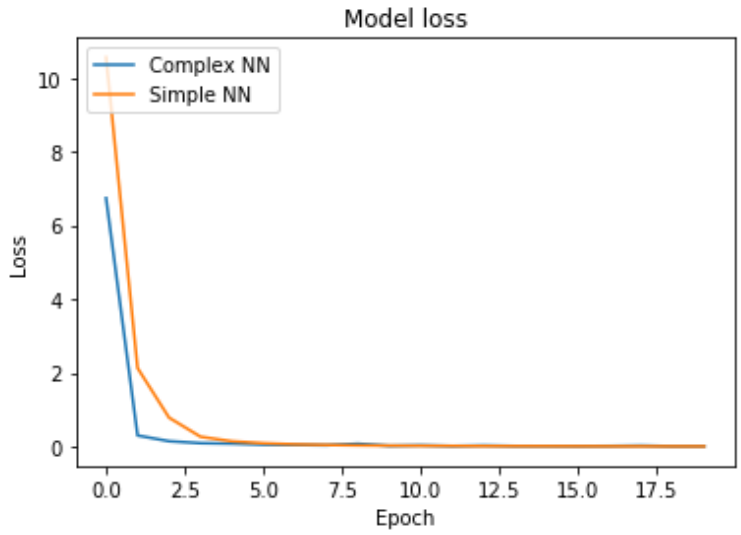

Figure 5.2: Comparing the 2 Networks training for the Point Load 1-Span Beam

We also tested on the data-set with the bending moments instead of deflection as the target. We trained the NN the same way as before and can observe an example of a prediction in Figure 5.4, proving our NN can also predict different functions of structure responses.

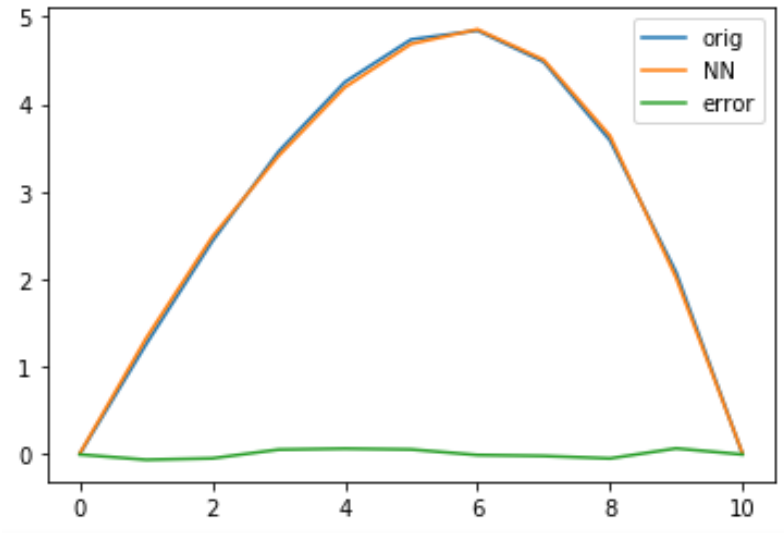

Figure 5.3: Predicting Deflection of Concentrated Load on 1-Span Beam

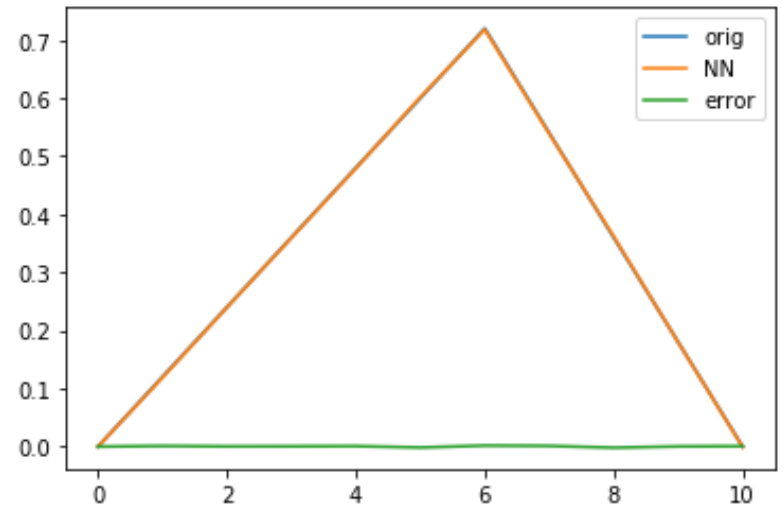

Figure 5.4: Predicting Bending Moment of Concentrated Load on 1-Span Beam

5.1.2 1-Span Beam Uniform Load

In the scenario of predicting 1-Span Beams when under a Uniform Load, the load is distributed along the beam, so we did not need 11 nodes to represent the position of the load, consequently our NN model was a little different.

The model used was similar to the first network for linear scenarios (figure 4.6), however only taking as input one node with the value of the size L of the Beam.

We can see that this simple Network was enough to predict the Deflection values accurately. The

loss plot in training can be observed in Figure 5.5 and an example of a prediction in Figure 5.6.

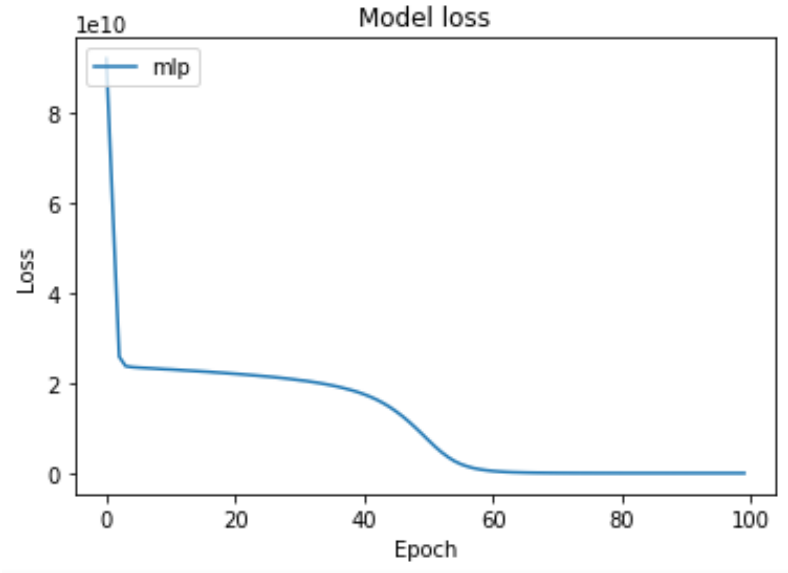

Figure 5.5: Training NN for Uniform Load on 1-Span Beam

Figure 5.6: Predicting Deflection of Uniform Load on 1-Span Beam

5.1.3 2-Span Beam Concentrated Load

This Scenario was the most complex of the Linear ones. We tested with both NNs in Figures 4.6 and 4.7, but with a slight difference in the input layer, since we needed to add more Nodes to represent the 2 spans now, instead of just one.

Comparison of the two Networks in training can be observed in Figure 5.7.

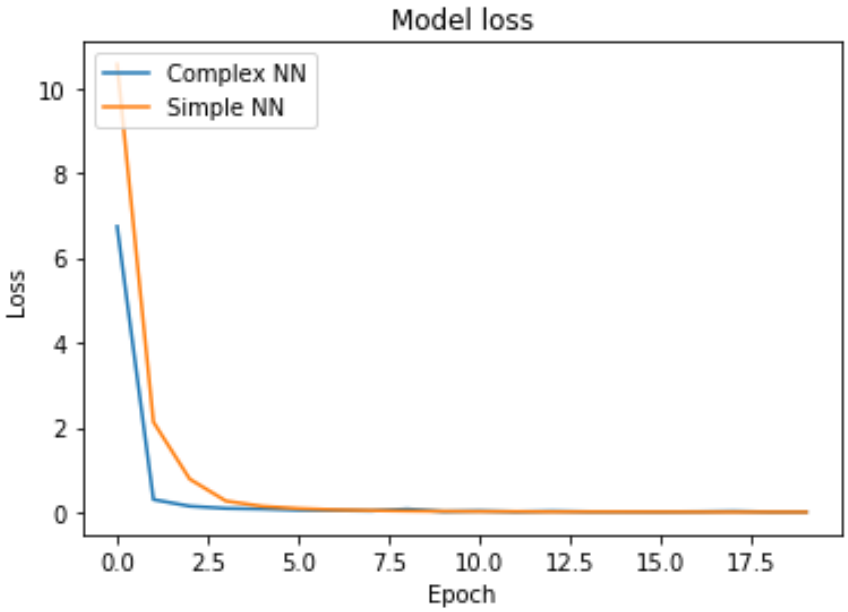

Figure 5.7: Comparing the 2 Networks training for the Point Load 2-Span Beam

We can then see that, again, the complex NN with the convolutional Layers (Figure 4.7) performed better, achieving the minimum error rate in less iterations. The reason behind this increased performance is probably exactly those convolution layers that converge better than normal dense layers when we give the nodes in a form representing the structure of our beam and therefore, the training is improved significantly.

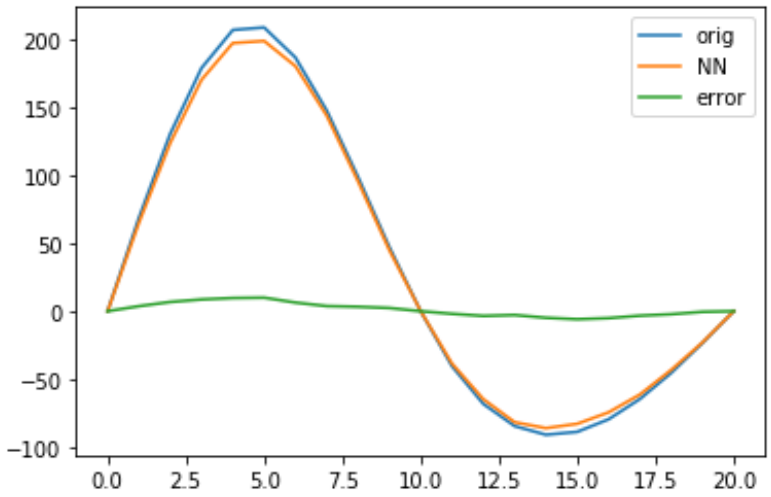

Figure 5.8: Predicting Deflection of Concentrated Load on 2-Span Beam

A prediction with this NN can be observed in figure 5.8. This scenario was the hardest to predict, the errors were slightly higher, however most predictions were very close to the original like in Figure 5.8.

Even though all these tests demonstrated were simple and solving these linear scenarios with traditional methods also takes a very short amount of time, so no real improvement is shown in these examples, it still serves as proof that Neural Networks are effective and capable of predicting very accurately deflections and bending moments of beams.

5.2 Nonlinear 2-Span Beams

5.2.1 Structure Responses Predictions

Onto the nonlinear scenario. The first test and comparison we should do was between the 2 Datasets with different types of loads obtained in ABAQUS. Compared to the real loads, the ABAQUS input loads were initially better normalized and did not have problems with multiple outputs for the same input. So we started by testing with this loads and observe the results.

With these loads, it was possible to correctly predict the Section Curvature (Figure 5.9), Bending Moment (Figure 5.10) and Rotation (Figure 5.11) values along the beam. As well as the maximum load supported by the beams (Figures 5.12 and 5.13).

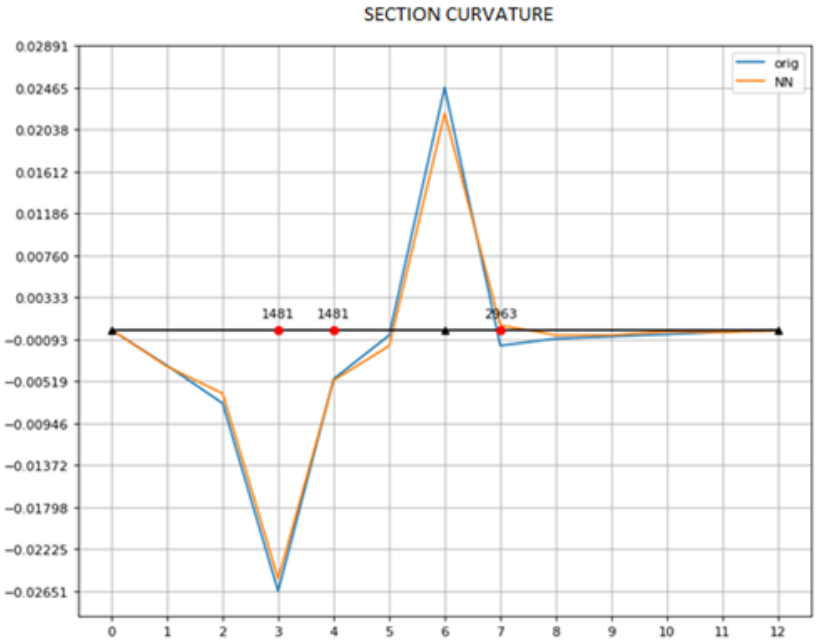

Figure 5.9: NN prediction of Section Curvature of nonlinear 2-Span Beam

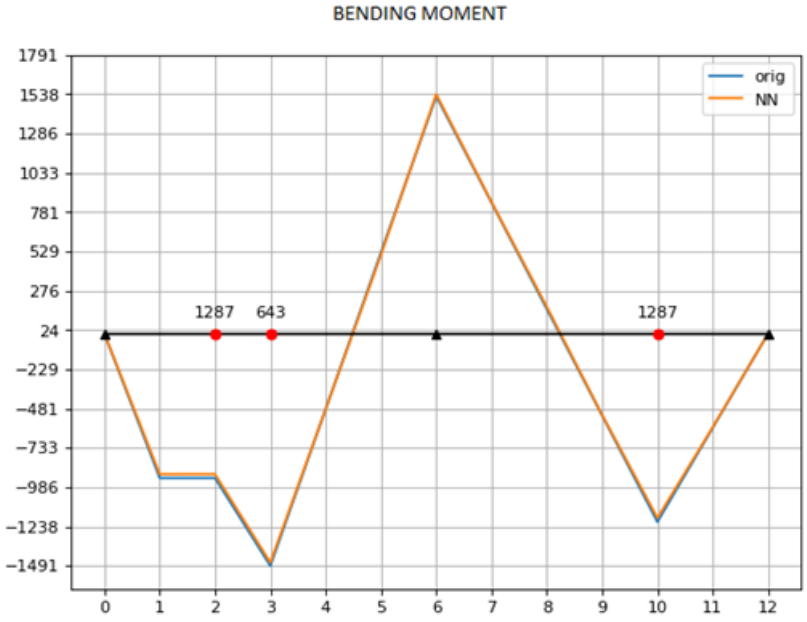

Figure 5.10: NN prediction of Bending Moment of nonlinear 2-Span Beam

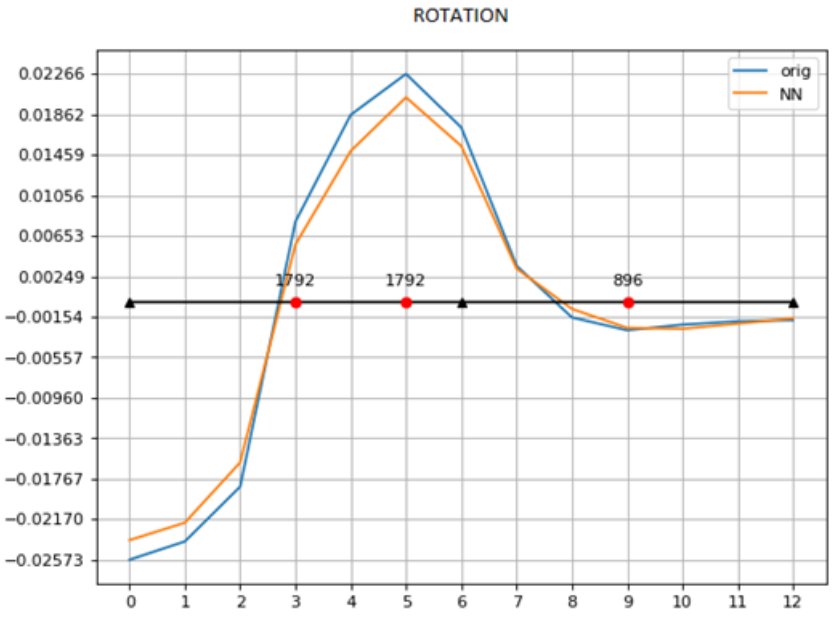

Figure 5.11: NN prediction of Rotation of nonlinear 2-Span Beam

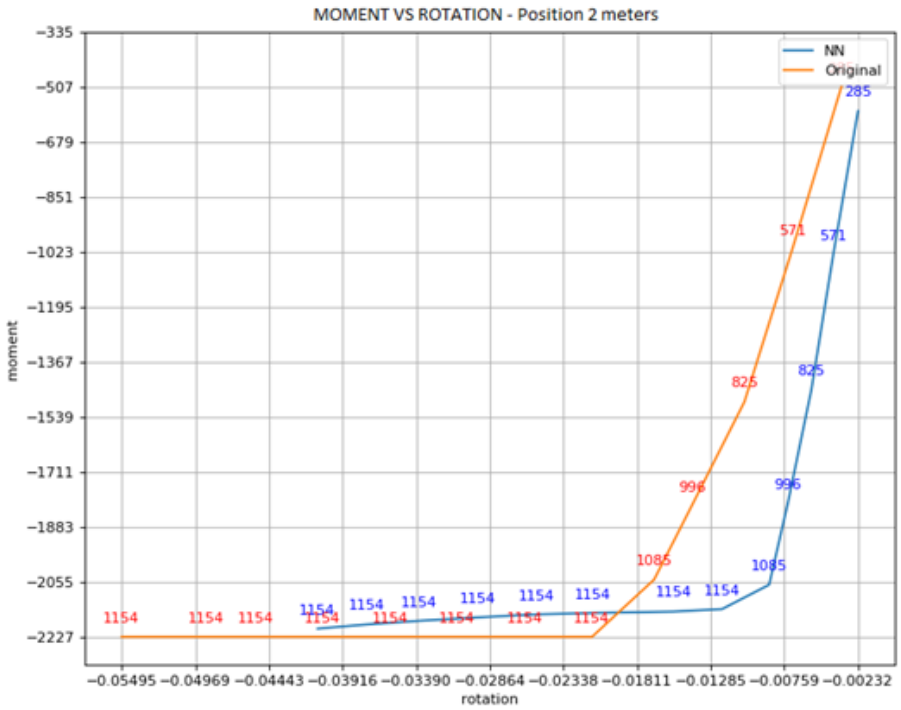

Figure 5.12: NN prediction of Moment vs Rotation of nonlinear 2-Span Beam in position 2 meters

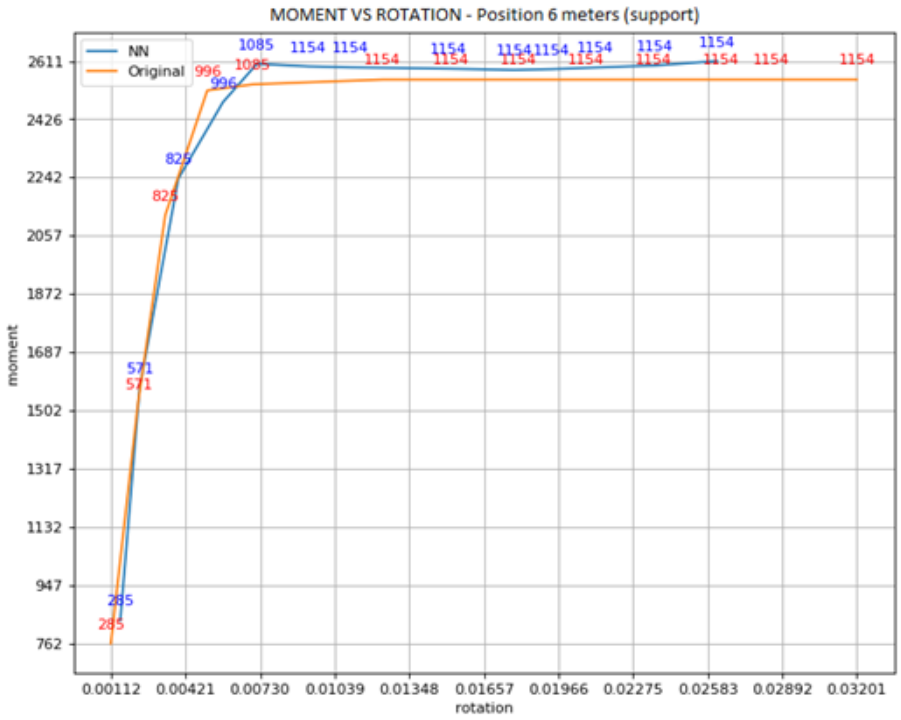

Figure 5.13: NN prediction of Moment vs Rotation of nonlinear 2-Span Beam in position 6 meters (support)

This shows how Neural Networks can effectively and accurately predict some already complex non-linear beam behaviours.

Using the real loads for the Section Curvature model was not possible due to the multiple samples with the same input corresponding to different outputs, as it was explained before and we wanted our model to be able to predict the real loads. So, for the Bending Moment predictions, we had to use the real loads. In this case not only it is possible, it is also better to use these loads. Consider the example of a fixed scenario, as we increase load, we reach the maximum and we keep inputting larger values of load in ABAQUS, but the result of Bending Moment is always the same since ABAQUS stopped at the maximum load supported. So, with the input loads dataset there will be a lot of samples for the same scenario but different loads that have the same result. Using the real loads computed at the end of the ABAQUS analysis, for the same scenario, a different value of load will always cause a different result, and this is better to model our Neural Networks. We can observe in Figure 5.14 the comparison in a thirty epochs training between these two types of loads.

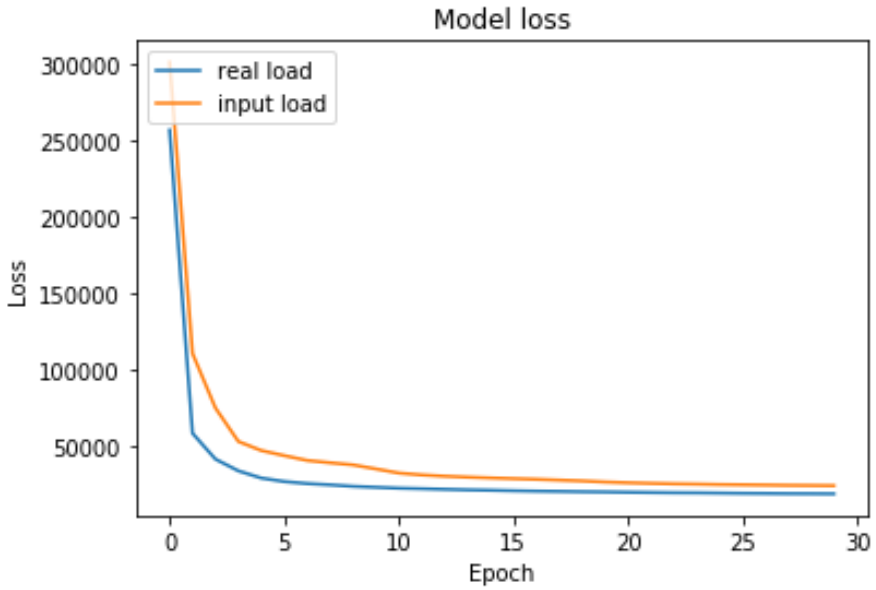

Figure 5.14: Comparison between the 2 load types when training for the Bending Moment

5.2.2 Networks comparisons

We developed 2 NN architectures (Figs. 4.8 and 4.9) to predict the non linear scenarios and compared them both in training and in predictions. Training was better with the NN architecture in figure 4.8 as we can observe the learning history comparison in Figure 5.15 however results were still quite accurate, since the final loss after training was close. We can see an example for the Section Curvature and Bending Moment predictions in Figures 5.16 and 5.17, respectively. We think that the use of Convolutional

Layers are a major factor in why the first network seems to outperform other architectures, since we can organize the information given to the network in a representation of the structure. This was also showed in the linear examples.

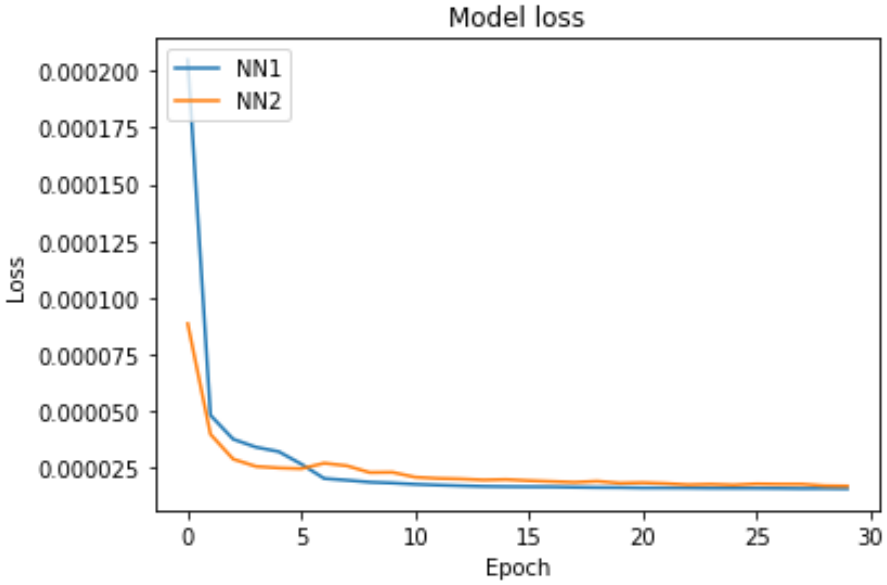

Figure 5.15: Comparison in training between the two NN architectures

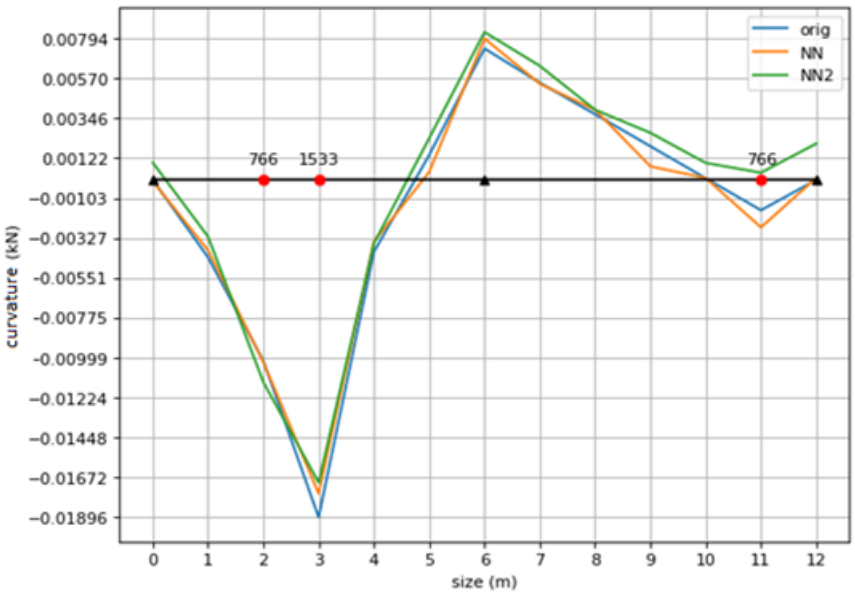

Figure 5.16: Comparison between the two NN architectures in predicting a Section Curvature example

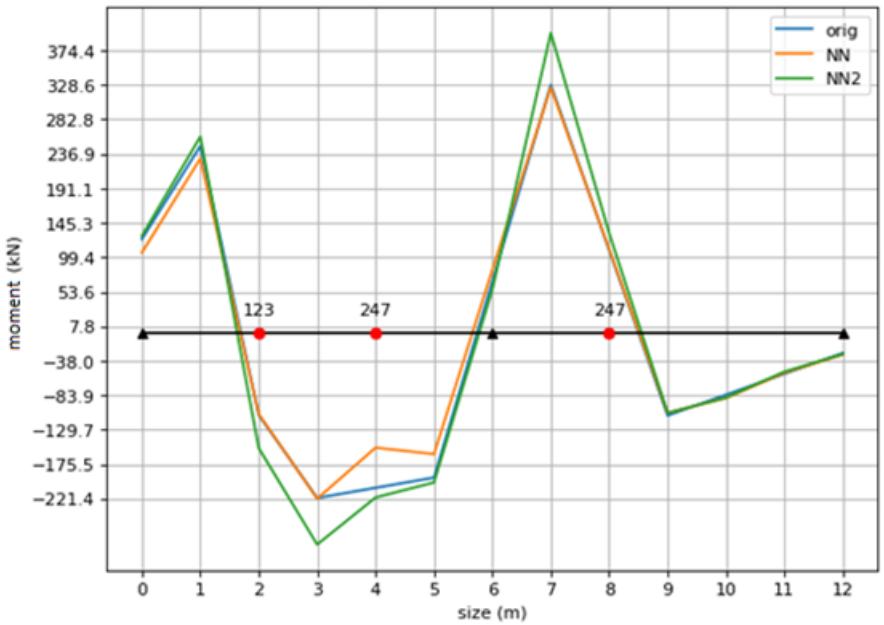

Figure 5.17: Comparison between the two NN architectures in predicting a Bending Moment example

5.3 Nonlinear 3-Span Beams

This scenario was the most complex of all the scenarios we studied here, because of being 3-span beams and because we varied types and magnitudes of loads as well as size of beams. There was lots of different possible examples inside this scenario and the solutions were nonlinear. We tested predictions on Rotation and Bending Moments responses as well as Section Curvatures. With this last one, we could identify the plastic hinges positions and then also predict some Rotation x Bending Moments graphs to identify the yield moment for the plastic hinge in cause, just like we did with the 2-span beams. Figure 5.18, 5.19 and 5.20 show a prediction of Rotation, Bending Moment and Section Curvature from random examples inside the domain of our scenario respectively. In Figures 5.21 and 5.22 we see the Rotation x Bending Moment graphs from two plastic hinges positions identified from the Section Curvature graph in figure 5.20.

As we can observe from the graphs, our model predicted rather accurately all this structure behaviours even though this scenario was a very complex and nonlinear scenario that would take several time to calculate with traditional methods like FEM.

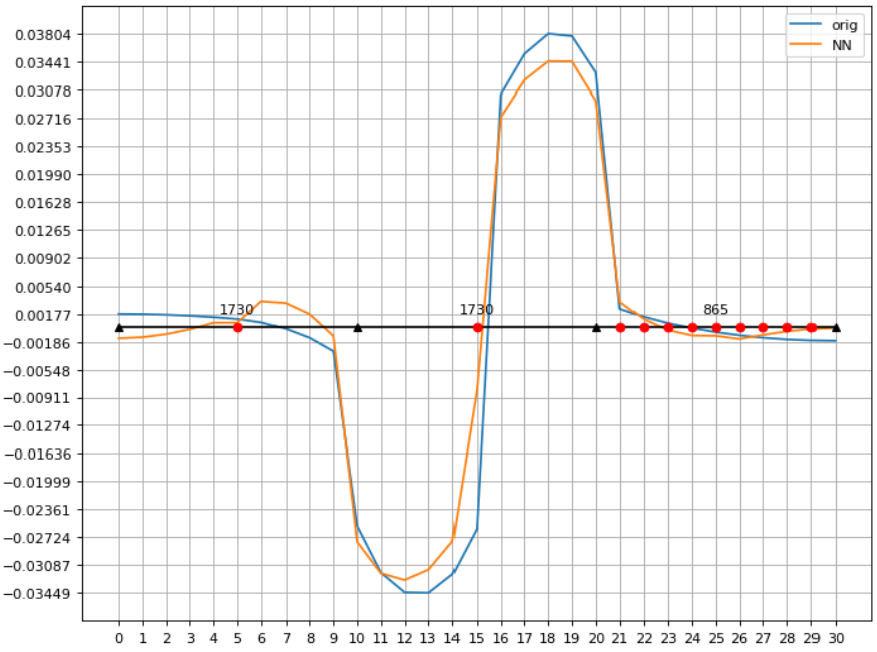

Figure 5.18: NN prediction of Rotation of 3-Span Beam, Sizes: 2m, 6m, 1m

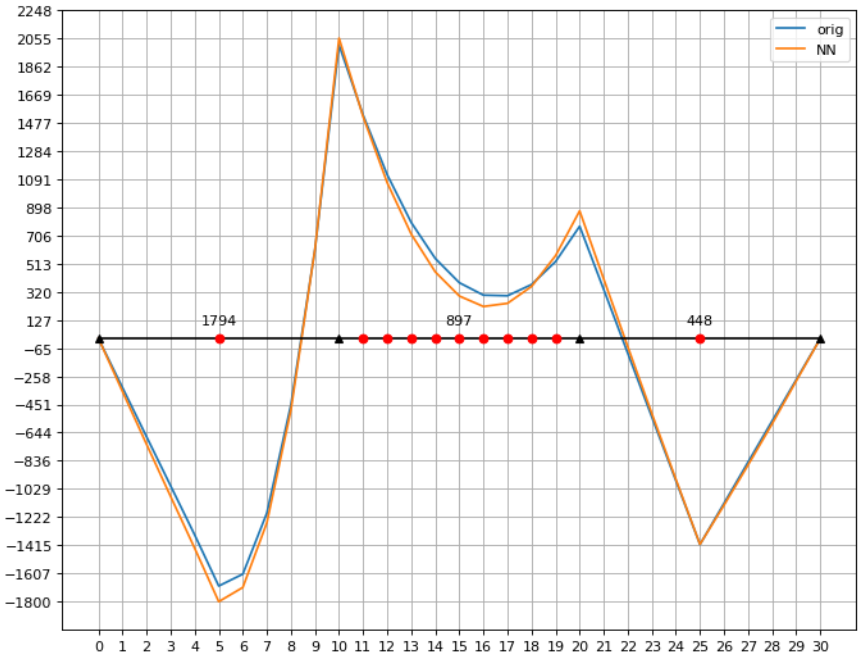

Figure 5.19: NN prediction of Bending Moment of 3-Span Beam, Sizes: 6m, 3m, 4m

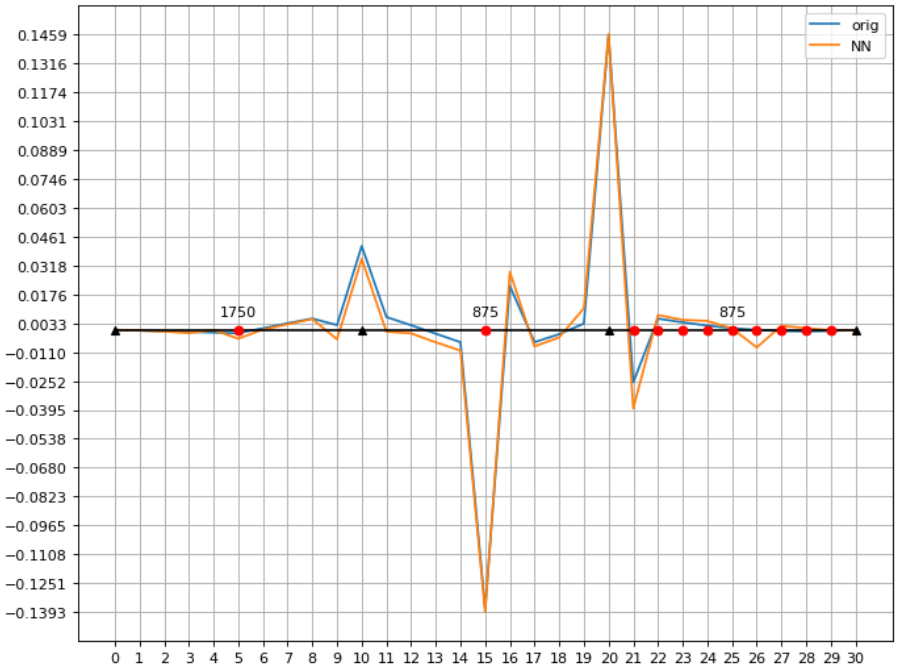

Figure 5.20: NN prediction of Section Curvature of 3-Span Beam, Sizes: 7m, 6m, 3m

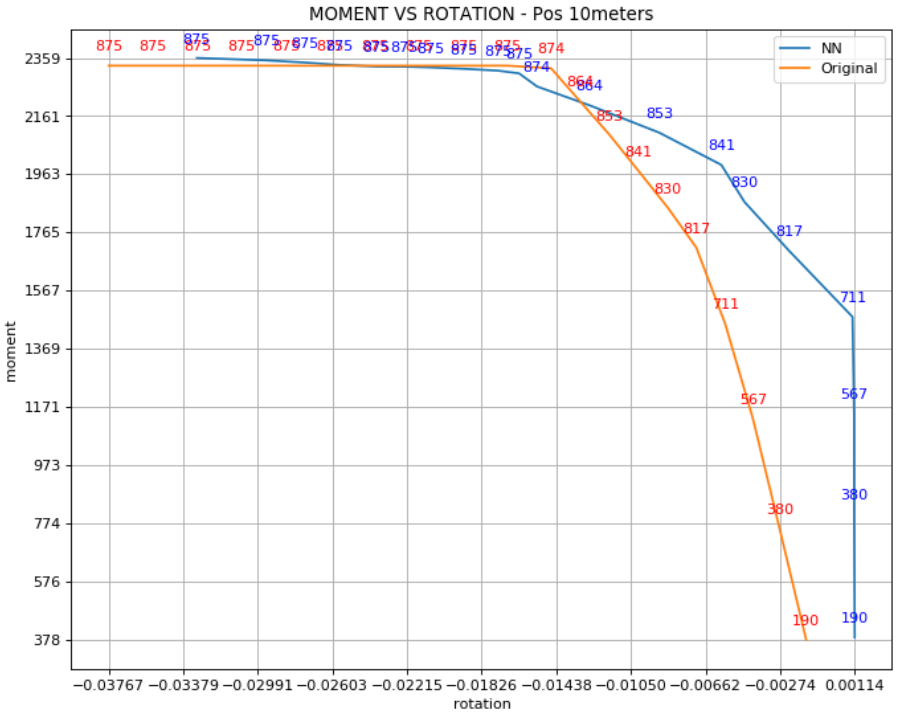

Figure 5.21: NN prediction of Bending Moment x Rotation of 3-Span Beam; Sizes: 7m, 6m, 3m; Position: 10m

5.4 Maximum Load Supported Predictions

To exemplify our method of finding out the maximum load supported by a beam we take an example of the 2-span beam scenario with $\alpha = \beta = 1$. It starts by plotting the section curvature graph when giving a

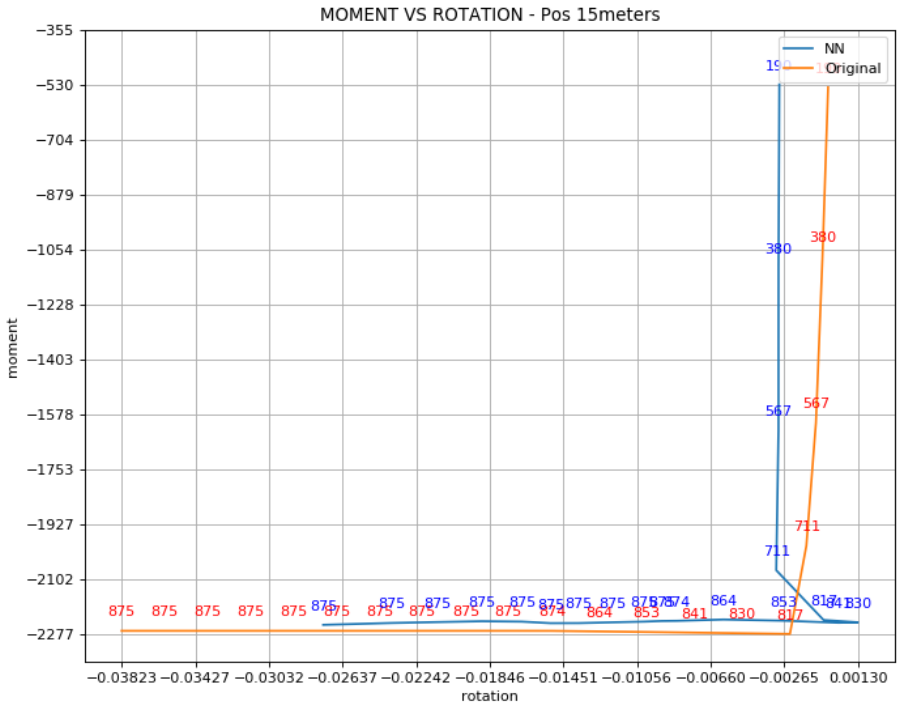

Figure 5.22: NN prediction of Bending Moment x Rotation of 3-Span Beam; Sizes: 7m, 6m, 3m; Position: 15m

value of load high enough to observe the plastic hinges points, as we can see in Figure 5.23 it would be positions 2 and 6 since those are the points where the section curvature value spikes.

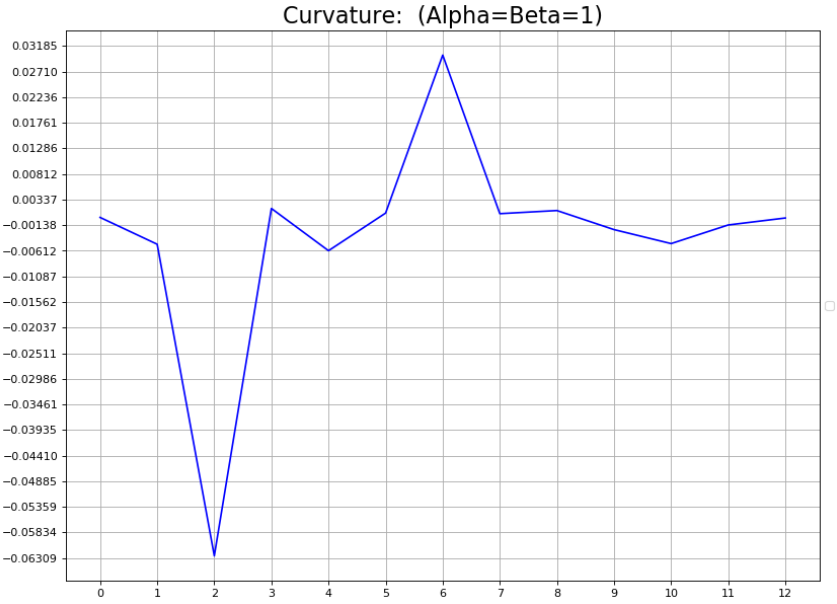

Figure 5.23: NN prediction of Section Curvature 2-Span Beam with α = β = 1

Then we calculate the rotation and bending moment graphs for the same scenario, multiple times

with incremented loads like in figures 5.24 and 5.25. We simulated the same beam with values of load from 25kN to 1975kN, increasing 50kN in each subsequent simulations. We did not go denser here, say 1kN in each simulation, for the sake of the clarity of the graphs to be showed in this document

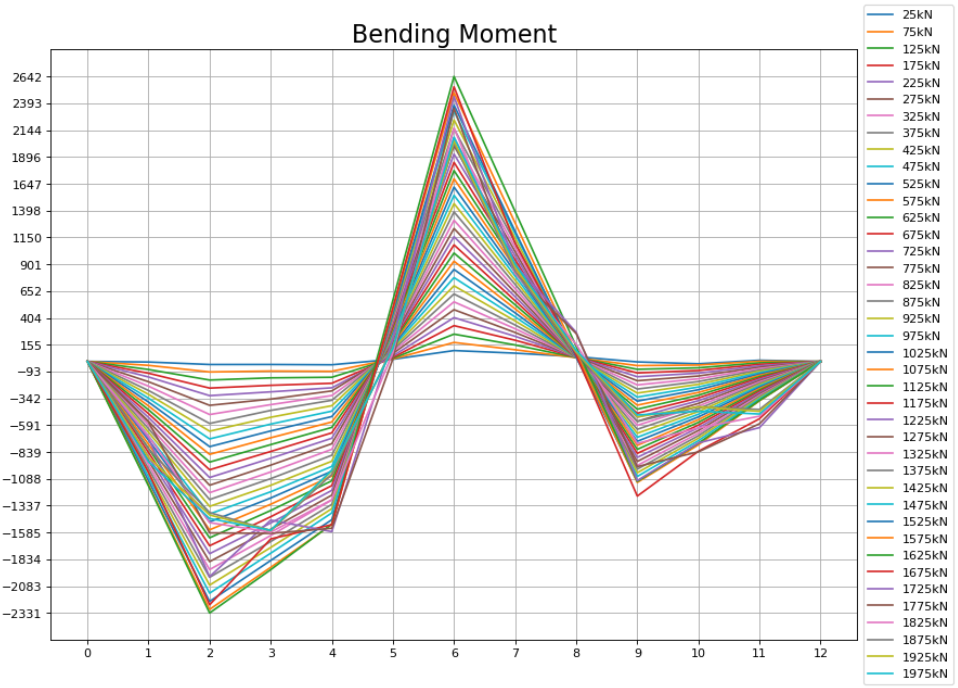

Figure 5.24: Multiple NN predictions of Bending Moment on 2-Span Beam with α = β = 1

And finally we can take the values of bending moment and rotation on the positions of plastic hinges identified before and plot them together to create the graph in blue in Figure 5.26, the orange graph in the same figure is the one computed by ABAQUS, where it will always stop increasing the load when reaching the maximum supported by the structure. This feature makes it very simple to calculate the maximum load supported by the beam, all there is to do is to observe the point in the graph where the slope of the function becomes 0. However, in our program that is not as easy, because we do not know beforehand the maximum load value. So, we will have to compute with values higher than the maximum load supported as is showed in Figure 5.26, but this values are not in our training data and are not even possible to exist, since our training data is made of exclusively examples computed by ABAQUS which never computes with values over the maximum load supported.

As we can observe in Figures 5.26, when the load surpasses the maximum load value, our neural network stops predicting in a behaviour similar to the real and starts giving out more random values. Some possible reason for this is because this values of load cannot be correctly represented in our training data since ABAQUS stops at the maximum, they exist because they are outliers in our training data computed by ABAQUS, most of this outliers have higher values of load but lower bending moment values than the maximum.

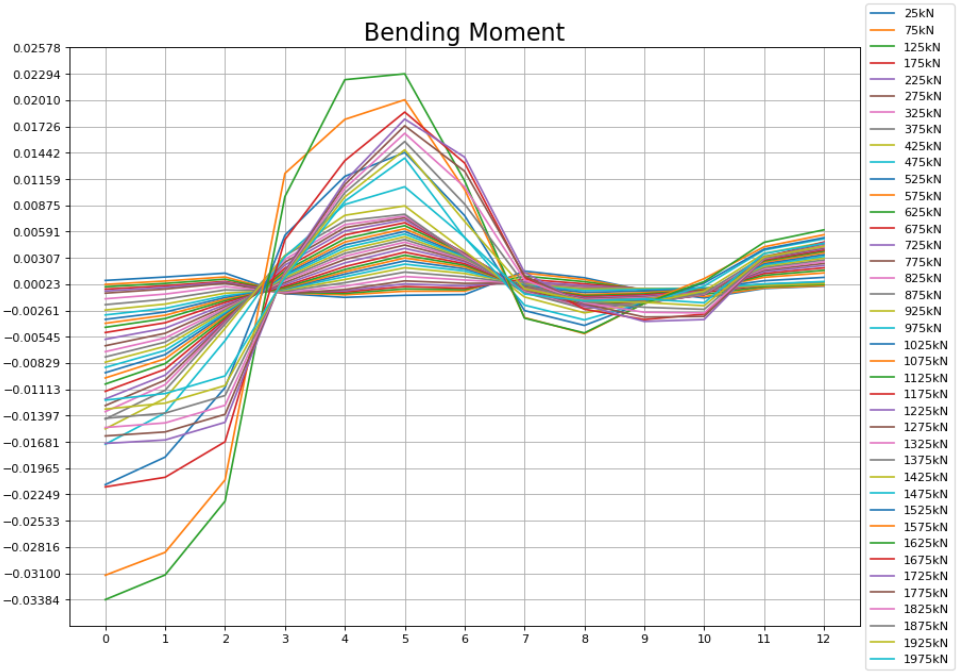

Figure 5.25: Multiple NN prediction of Rotation on 2-Span Beam with $\alpha = \beta = 1$

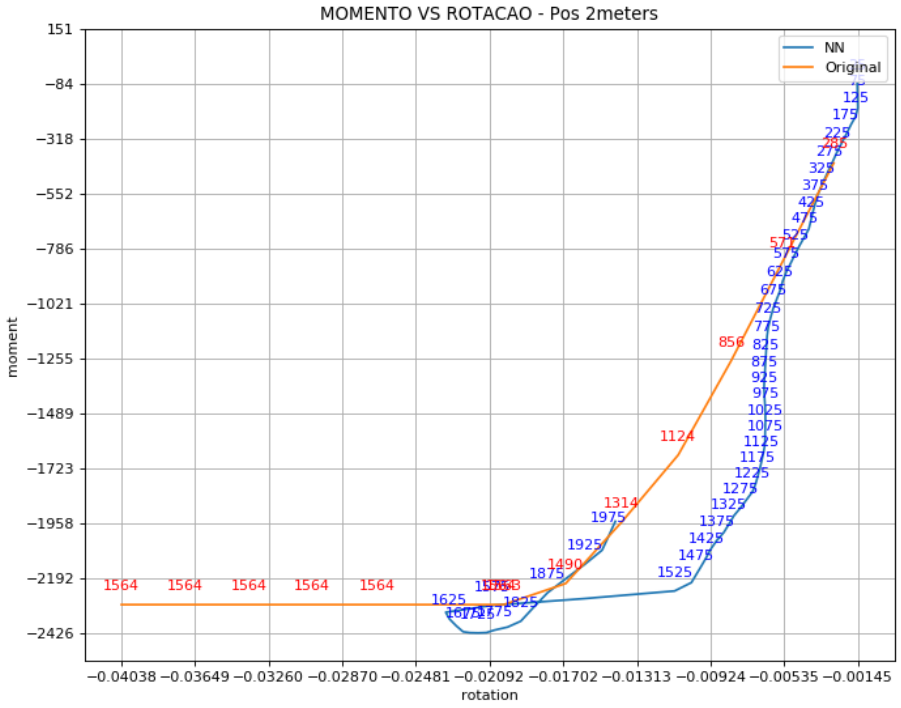

Figure 5.26: Comparison between real and NN prediction of Moment vs Rotation of 2-Span Beam with $\alpha = \beta = 1$

Given all this and observing the graphs with the results, our method to identify the maximum possible load was the last point before the absolute value of bending moment lowers, that is, the last point before

the function starts having a different/more random behavior. We can observe some examples in Figures 5.27, 5.28 and 5.29 that show that this method predictions are already quite accurate, but we have a sparse number of increments, for the sake of the clarity of these plots we increment the load by 50kN each step. In our final model our incrementations are denser, we increment load by just 1kN in each step, leading to even more accurate predictions.

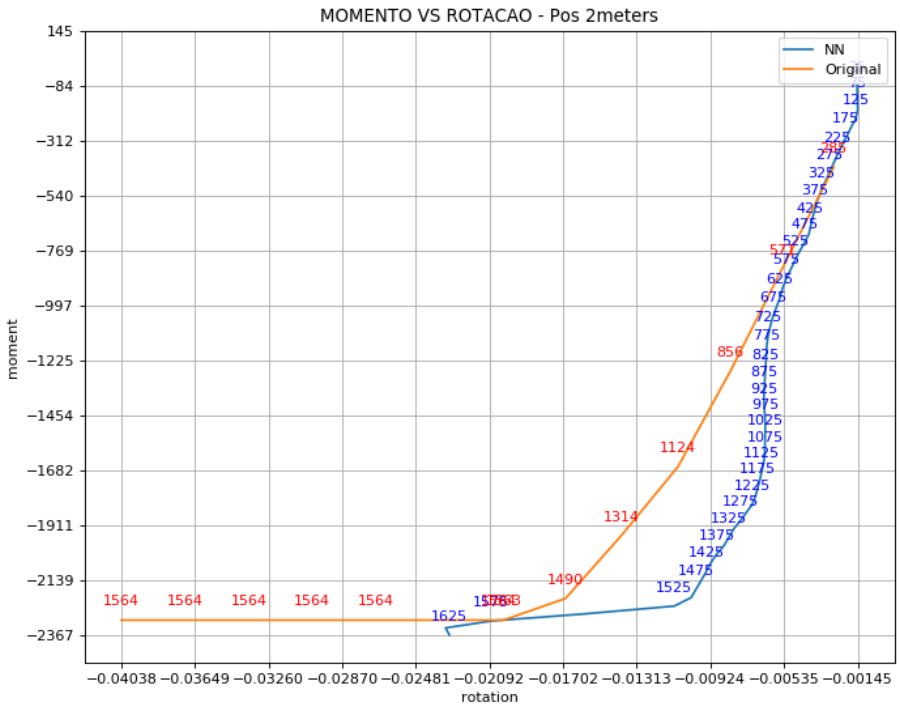

Figure 5.27: Comparison between real and Final Model prediction of Moment vs Rotation of 2-Span Beam with α = β = 1

In the example of Figure 5.27 the Neural Network would predict a maximum load of 1625kN when the original is 1564kN, in Figure 5.28 NN would predict 1175kN when original is 1137kN, and in Figure 5.29 NN would predict 1975kN when original is 1927kN. These results are good estimated values that are quickly computed and could be used for quickly designing multiple different nuances of a beam scenario.

With this method of identifying the maximum load supported, we only had to predict the bending moment, since we only need to observe the tweak in the Bending Moment value to identify the instant where the maximum load was reached, therefore the rotation model although demonstrated in this work with having good predictions was not used in our final model.

Our program simulates around three thousand different incremented loads for a given scenario, then computes and outputs the result in around four and a half seconds. To have a notion of the potential of Neural Networks and how much faster our model is compared to traditional methods, when we simulated, for the same scenario, the thirteen different load increments, represented in Figures 5.12 and 5.13,

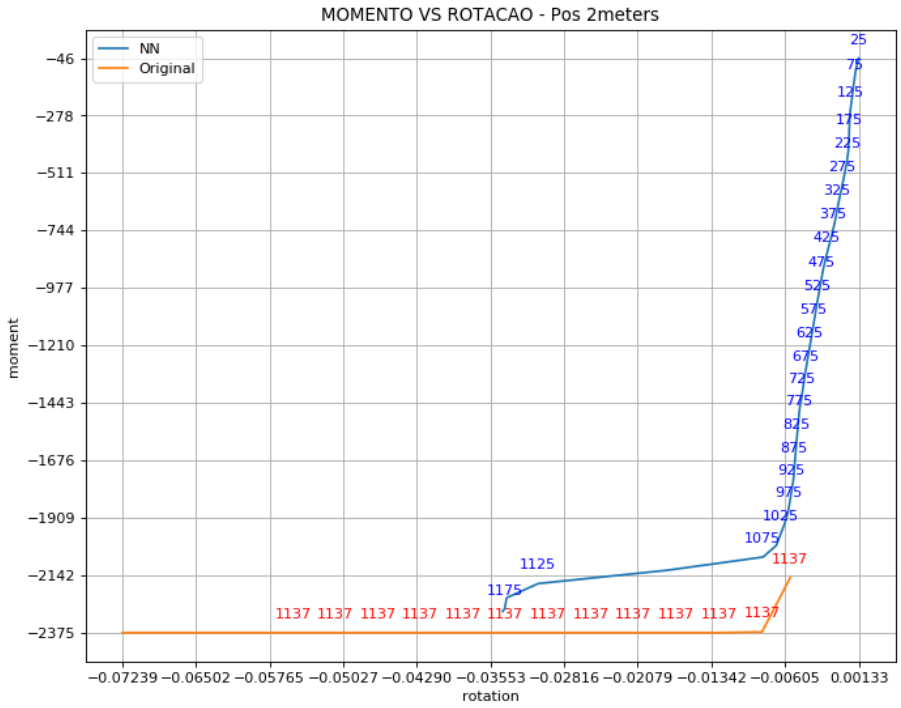

Figure 5.28: Comparison between real and Final Model prediction of Moment vs Rotation of 2-Span Beam with α = β = 1

just the ABAQUS simulations before plotting them all together took 325 seconds. The Neural Network models computed and plotted the same thirteen samples in 0.007 seconds.

The following table shows some scenarios executions times comparisons between ABAQUS FEM based simulation and our NN model predictions.

Table 5.1: Beam Scenarios execution times

Scenario	Nr of simulations	ABAQUS	NN
Linear 2-Span Beam	1	27s	0.001s
Linear 2-Span Beam	20	541s	0.002s
Non Linear 2-Span Beam	1	27s	0.001s
Non Linear 2-Span Beam	20	541s	0.005s
Non Linear 3-Span Beam	1	28s	0.001s
Non Linear 3-Span Beam	20	565s	0.005s

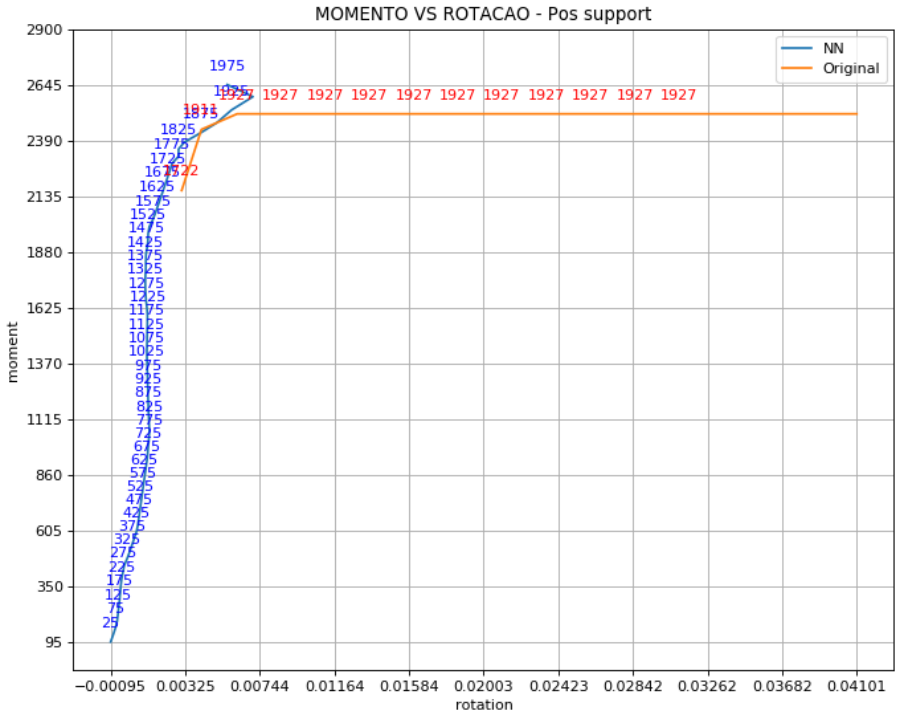

Figure 5.29: Comparison between real and Final Model prediction of Moment vs Rotation of 2-Span Beam with α = β = 1

6

Conclusion

Contents

6.1 Conclusions

Planning and building a civil engineering structure is a serious and complex project, designing a safe architecture for the structure is of course an essential part to avoid risks. Nowadays, conventional methods to verify the safety of a complex and non linear structure can be very time expensive. Neural Network approaches can help to reduce this computational time.

We defined 3 scenarios in linear models and 2 scenarios in non linear models to work on.

Table 6.1: Beam Scenarios tested

Scenario	Material
1-Span Beam Uniform Load	Linear
1-Span Beam 1 Concentrated Load	Linear
2-Span Beam 1 Concentrated Load	Linear
2-Span Beam 3 Concentrated Loads	Nonlinear
3-Span Beam 2 Concentrated Loads 1 Uniform Load	Nonlinear

We believe to have developed neural network architectures build exceptionally to predict inside these scenarios, even though some room for improvement is always possible, we still showed aspects that seem to be the most optimal such as the superiority in training of ReLu compared to any other AF, and Adam compared to any other optimizer, as well as our developed architecture that uses convolutional layers to better model the real world structure.

With this work we proved the potential of neural network models for this specific problem by building a model that can predict an approximation of a solution for some types of beam structures in 30 000 less time than with a traditional method. This model can also, with more training data, evolve into a more multifaceted model that can predict different designs of structures.

6.2 Future Work

6.2.1 Generalization

One thing to improve in our models is the capacity to generalize more different and diverse beam designs, even though our model can predict with high accuracy, it fails to generalize, because it can only yet predict beam architectures of 3 spans at maximum.

Obtaining new training data to incorporate different designs, and then test the prediction results on them, would be the first step to improvement. For this it would be necessary to obtain more data of different structure designs and develop more NN architectures to represent the new scenarios. Then have some sort of selection method to choose which NN to use in each scenario.

6.2.2 Incorporation inside FEM

Another interesting goal for the project initiated here would be to incorporate it inside the FEM model. FEM has a computational expensive method for solving the structure responses of non linear models, our model offers an alternative fast approximation of this responses for an easier and faster modeling at a preliminary design phase. However, it could also be possible to incorporate our model with the FEM to replace the more complex and computational expensive functions with the NN. This would be even more effective after an improvement to our model's generalization capacity, making possible to improve FEM computational time efficiency with any structure model at any phase of a structure creation.

6.2.3 Solving PDE directly

This would not be an improvement to be made to our model, but more of an alternative with higher potential to solve the same problem. We did not test for this approach in this dissertation, however it is a possible solution to our initial objective that also involves neural networks. As reviewed in Chapter 2.3, we could use the neural network models to learn and predict the partial differential equations solutions that have to be solved for FEM in non linear models that are computational expensive and significantly increase the time efficiency. This would be even more interesting if the model was then incorporated inside softwares that use FEM, such as ABAQUS.

Bibliography

[1] H. Salehi and R. Burgueno, "Emerging artificial intelligence methods in structural engineering," *Engineering structures*, vol. 171, pp. 170–189, 2018.

[2] S. Lee, J. Ha, M. Zokhirova, H. Moon, and J. Lee, "Background information of deep learning for structural engineering," *Archives of Computational Methods in Engineering*, vol. 25, no. 1, pp. 121–129, 2018.

[3] R. Lopez, E. Balsa-Canto, and E. Oñate, "Neural networks for variational problems in engineering," *International Journal for Numerical Methods in Engineering*, vol. 75, no. 11, pp. 1341–1360, 2008.

[4] S. Chaudhary, U. Pendharkar, and A. K. Nagpal, "Bending moment prediction for continuous composite beams by neural networks," *Advances in Structural Engineering*, vol. 10, no. 4, pp. 439–454, 2007.

[5] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity." 1943.

[6] L. Fausett, "Fundamentals of neural networks architectures, algorithms and applications," 2008.

[7] J. Han, A. Jentzen, and E. Weinan, "Solving high-dimensional partial differential equations using deep learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 34, pp. 8505–8510, 2018.

[8] M. Kaczmarek and A. Szymańska, "Application of artificial neural networks to predict the deflections of reinforced concrete beams," *Studia Geotechnica et Mechanica*, vol. 38, no. 2, pp. 37–46, 2016.

[9] C.-M. Chang, T.-K. Lin, and C.-W. Chang, "Applications of neural network models for structural health monitoring based on derived modal properties," *Measurement*, vol. 129, pp. 457–470, 2018.

[10] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks."

[12] E. Fathalla, Y. Tanaka, and K. Maekawa, “Remaining fatigue life assessment of in-service road bridge decks based upon artificial neural networks,” *Engineering Structures*, vol. 171, pp. 602–616, 2018.

[13] J. Shu, Z. Zhang, I. Gonzalez, and R. Karoumi, “The application of a damage detection method using artificial neural network and train-induced vibrations on a simplified railway bridge model,” *Engineering structures*, vol. 52, pp. 408–421, 2013.

[14] K. H. Padil, N. Bakhary, and H. Hao, “The use of a non-probabilistic artificial neural network to consider uncertainties in vibration-based-damage detection,” *Mechanical Systems and Signal Processing*, vol. 83, pp. 194–209, 2017.

[15] A. Al-Aradi, A. Correia, D. Naiff, G. Jardim, and Y. Saporito, “Solving nonlinear and high-dimensional partial differential equations via deep learning,” *arXiv preprint arXiv:1811.08782*, 2018.

[16] J. B. Pedro, J. Maroñas, and R. Paredes, “Solving partial differential equations with neural networks,” 2019.

[17] V. Chandwani, N. K. Gupta, R. Nagar, V. Agrawal, and A. S. Jethoo, “Artificial neural networks aided conceptual stage design of water harvesting structures,” *Perspectives in Science*, vol. 8, pp. 151–155, 2016.

[18] D. Lehkỳ, O. Slowik, and D. Novák, “Reliability-based design: Artificial neural networks and double-loop reliability-based optimization approaches,” *Advances in Engineering Software*, vol. 117, pp. 123–135, 2018.

[19] M. Stoffel, F. Bamer, and B. Markert, “Artificial neural networks and intelligent finite elements in non-linear structural mechanics,” *Thin-Walled Structures*, vol. 131, pp. 102–106, 2018.

[20] J. Ghaboussi, C. J, and X. Wu, “Knowledge-based modeling of material behavior with neural networks,” *Engineering Mechanics Division*, vol. 17, pp. 32–153, 1991.

[21] V. Papadopoulos, G. Soimiris, D. Giovanis, and M. Papadrakakis, “A neural network-based surrogate model for carbon nanotubes with geometric nonlinearities,” *Computer Methods in Applied Mechanics and Engineering*, vol. 328, pp. 411–430, 2018.

[22] A. Javadi, M. Mehravar, A. Faramarzi, and A. Ahangar-Asr, “An artificial intelligence based finite element method,” *Computers and Intelligent Systems*, vol. 1, no. 2, 2009.

[23] Z. Tadesse, K. A. Patel, S. Chaudhary, and A. K. Nagpal, “Neural networks for prediction of deflection in composite bridges,” vol. 68, no. 1, pp. 138–149, 2012.

[24] J. Sirignano and K. Spiliopoulos, “Dgm: A deep learning algorithm for solving partial differential equations,” *Journal of Computational Physics*, vol. 375, pp. 1339–1364, 2018.

[25] E. Weinan and B. Yu, “The deep ritz method: a deep learning-based numerical algorithm for solving variational problems,” *Communications in Mathematics and Statistics*, vol. 6, no. 1, pp. 1–12, 2018.

[26] T. I. Aksyonova, V. V. Volkovich, and I. V. Tetko, “Robust polynomial neural networks in quantative-structure activity relationship studies,” *Systems Analysis Modelling Simulation*, vol. 43, no. 10, pp. 1331–1339, 2003.

[27] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations,” *arXiv preprint arXiv:1711.10566*, 2017.

[28] ——, “Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations,” *arXiv preprint arXiv:1711.10561*, 2017.