Development of a Deformation-Based Structural Health System with Contactless Sensors and Machine Learning for Health Characterization and Failure Prediction

Juan Camilo Avendano Arbelaez

Development of a Deformation-Based Structural Health System with Contactless Sensors and Machine Learning for Health Characterization and Failure Prediction

by

Juan Camilo Avendano Arbelaez

A dissertation submitted to the College of Engineering and Science of
Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Systems Engineering

Melbourne, Florida
December, 2020

We the undersigned committee hereby approve the attached dissertation,
"Development of a Deformation-Based Structural Health System with Contactless Sensors and Machine Learning for Health Characterization and Failure Prediction"
by
Juan Camilo Avendano Arbelaez

Luis Daniel Otero, Ph.D.
Associate Professor
Computer Engineering and Sciences
Major Advisor

Ersoy Subasi, Ph.D.
Assistant Professor
Computer Engineering and Sciences
Committee Member

Aldo Fabregas Ariza, Ph.D.
Assistant Professor
Computer Engineering and Sciences
Committee Member

Munevver Mine Subasi, Ph.D.
Associate Professor
Mathematical Sciences
Committee Member

Philip Bernhard, Ph.D.
Associate Professor and Department Head
Computer Engineering and Sciences

Abstract

Title: Development of a Deformation-Based Structural Health System with Contactless Sensors and Machine Learning for Health Characterization and Failure Prediction

Author: Juan Camilo Avendano Arbelaez

Advisor: Luis Daniel Otero, Ph.D.

This dissertation presents the design and development of a structural health monitoring (SHM) system specifically tailored for transportation infrastructure components, such as bridges. The proposed system collects data by using contactless sensors and performs health characterization and failure prediction. It is capable of simulating multiple load conditions on structures, identifying possible failure points, and detecting and predicting failure scenarios. Both hardware and software implementations of a model of a bridge were performed as a pilot project in order to validate the proposed system. Computer simulation in ANSYS and the application of gradient boosting neural networks were performed to produce a comparative and predictive analysis of the behavior of transportation infrastructures, which can be used to understand the health of the structure and make informed decisions.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank:

My advisor, Dr. Luis Daniel Otero, for his help, guidance, and support during the research work presented here. Without him, it would have been impossible to complete this project. He has given me invaluable guidance throughout my research and academic career.

My committee – Dr. Munevver Subasi, Dr. Ersoy Subasi and Dr. Aldo Fabregas Ariza - for their invaluable advice.

My friends - Manuel Jaramillo, Deep Patel, Darshan Yadav, David Beavers, and Matias Fernandez - whose help was instrumental to this work.

A special thanks to Ms. Daphne Otarola for her support and motivation throughout this process.

Moreover, I must thank my family, specifically my parents, Luis J. Avendano and Silvia Arbelaez, my brother, Jose N. Avendano, and my uncles, grandparents, and cousins, for encouraging me to continue moving forward. Without their support and sacrifices, this research work would have not have been possible. I am lucky to have the support of many people in numerous ways.

Dedication

To my family,

Because all of our achievements are fruits of our collective effort.

Chapter 1 Introduction

Problem Statement

Several regions across the United States have expressed concerns over the alarming status and failure rate of key transportation infrastructure components such as large bridges [1], [2], [3]. Studies of similar structures show that health information and failure prediction of bridges rank as one of the top priorities of the United States [4], [5]. Given the vast amount of small to medium-sized structures currently in place in the national transportation system, it is essential to expand the state-of-the-art technology in this field by developing cost-effective structural health monitoring (SHM) methods that allow for the continuation of service during the study.

Motivation

Addressing the aforementioned requirements, funded research projects were initiated with the objectives of implementing dynamic SHM programs, predicting the actual behavior of bridge structures and designing a long-term instrumentation plan [6]. As monitoring the performance of structures plays a crucial role in identifying the risk of their failure, the development of innovative SHM systems has become an emerging field of research [7]. With technological advancements, studies in the inspection and calculation of structural health data via different types of sensor systems have evolved significantly in the past few years [8]. On the one hand, fiber optics sensors are common and practical, as they are highly reliable and can be embedded in large concrete structures during the construction phase [9]. On the other hand, the deployment and the high cost of sensor arrays could be (or have been) significant obstacles. Due to the complexities in deployment, the use of fiber optical systems in small-to-medium-size structures, such as small bridges and traffic signal mast arms (TSMA), is not a cost-effective solution. Fiber optical systems are cost-effective when used in large complex structures, however, because the expected lifespans of these structures are longer [10]. Yet, wide-scale adoption of fiber optic deployment on large bridges seems unfeasible. Installation would cause large-scale traffic interruptions while the fiber is embedded in the deck and while structural modifications are made through drilling, cutting surface patches, wiring, and so forth.

Other types of sensors such as extensometers, vibration detection devices, and strain gauges are also plagued by cumbersome deployment and installation procedures. An investigation of influential parameters (amplitude, spectral content of the dynamic displacements, location and orientation of sensors, fusing inputs, noise effects, rolling-shutter effects, etc.) that are required to be considered in the selection of sensors for structural dynamic applications is presented in [11]. This focuses on inexpensive digital cameras with depth-sensing capabilities which were proven suitable for measurement of a 3D dynamic displacement-field. A completely contactless SHM system framework, which was developed by using regular cameras and computer vision techniques for detecting displacements and vibrations of structures, is presented in [12]. A review of studies in the field concludes that vision-based systems outperform traditional displacement sensors in terms of instrumentation cost, installation efforts, and measurement capacity [13]. Therefore, a non-contact sensor system that uses infrared wavelengths was used in this research.

Recent advancements in computational power have allowed for the inclusion of new techniques for SHM. The adaptation of predictive algorithms facilitates the collected data's extrapolation to identify possible structural failures. The use of machine learning technology within the realm of artificial intelligence has spawned the development of neural network (NN) architectures for predicting structural failure modes. Modeling of structures using artificial NNs for damage detection is presented in the literature [14].

A NN is an artificial intelligence system designed to recreate a biological model of nervous systems [15]. The advantage of NNs over the other algorithms is their ability to learn from the embedded data. During the NN's training process, it finds special parameters hidden from human perception [16]. As a result of the training, the network can improve its performance over time according to certain rules. When comparing the NN approach with statistical methods, some differences can be identified. Although statistical techniques, such as regression analysis, learning algorithms and so forth are widely used in SHM, they consume more time and effort and are probabilistic in nature [14]. The use of unverified technologies causes errors in monitoring and increases the risk of emergencies whose consequences can lead to human casualties. Moreover, accidents and disasters bring not only severe moral and social shocks, but also financial losses due to the inoperability of the object and the need for restoration. Due to these reasons, NNs become an indispensable tool for SHM. It is a compelling modeling method that allows reproduction of extremely complex dependencies, particularly ones that are non-linear. Hence, several machine-learning mechanisms, such as gradient boosting, were tested for suitability for this research, and gradient boosting was employed in order to identify and predict structural health.

Research Objectives

The objectives of this research were based on current needs and the problem statement. The problem statement can be summarized as follows: to find an effective, non-invasive, and low-cost solution for structural health analysis and the prognosis of transportation structures, such as bridges.

After analyzing and identifying the literature related to the subject of advanced SHM, it was clear that elements of this system can be used for processes such as tracking a failure point within a structure, and understanding and comparing the reaction to expected loads in relationship with experimental and simulated loads. Furthermore, there is a need for a system that facilitates characterization of structural health, as well as predicts its future state(s). As per the research outcomes published thus far, most of the sensor arrays demand special installation requirements, including embedded fabrics and other contact points. They are hard to implement in larger structures or geographically challenging points (for example, tall water bridges) and require added expenses for such installations.

The primary focus of this work is to develop an effective structural health prediction system that utilizes advanced artificial intelligence technologies and cost-effective sensors. The main research objectives are listed as follows:

1. Identify necessary sensor data and sensor equipment to understand the structural health and behavior of structures.

2. Identify an effective placement of sensors or sensor nodes.

3. Develop a structural modeling framework to simulate multiple structural loads.

4. Analyze multi-sensor data and incorporate it into a reliable prognosis system.

The system characteristics are determined by identifying the most critical elements in material failure of the infrastructure of interest, as well as the available sensor technology. The proposed system operates by using remote sensing technology for the monitoring of the structure and the measurement of deformation at multiple points. The system suggests incorporating a mathematical programming model for optimal sensor placement, a modeling framework for SHM to create training datasets, and a library of known behavior patterns. The primary model can generate a library of failure and non-failure training data at an infinite number of loading conditions by using the available structural model.

The training data generated for this research study is based on currently known loading factors for civil engineering infrastructures, such as bridges. Once the loading conditions are identified and characterized, they are fed into the computer model to generate a data set of expected stress and strain in the structure. From the computer model, the ideal values for the deformation of the structure under these loading conditions can be obtained. Moreover, any point in which the material passes its elastic deformation or any material failure can be identified.

The generated library is used to develop the machine learning algorithm, so that it can detect necessary correlations of input data with known loading conditions and failure modes. Additionally, it will extrapolate input data to match and identify the possible relationship to failure scenarios. The neural network will use any input data outside the original training library as training points, improving accuracy and predictive capacity over time based on the number of cases introduced to the network.

The result is a computational framework for the study of failure modes in bridge structures, which includes the optimal placement of a surface mounted sensor array. Furthermore, a comparable data cloud of sensor readings will be developed for each of the most frequently reoccurring failure modes. It will facilitate expanding the research to deploy a prototype sensor array and compare live data to the simulated point cloud for each failure mode. This comparison will provide an indication of possible failure modes and the expected life of the structure.

Chapter 2 Literature Review

Structural Health Monitoring (SHM)

As per the literature, there are numerous definitions for SHM. According to [17], SHM refers to a monitoring system that can acquire and process data to evaluate structural health for damage detection and prognostics. The integration of automated health assessment analytics distinguishes SHM systems from traditional monitoring systems. The first crucial stage of an SHM system is data acquisition or data generation, as this step determines the success of subsequent steps. Article [18] discusses various methods of SHM data gathering and their importance. Usually, the structure's response data is obtained through experimental measurements using sensors like linear variable displacement transducer (LVDT), accelerometers, and many more. Data acquisition typically happens on lab-scale substitutes due to practical constraints. However, data can also be generated computationally via numerical methods, such as the finite element (FE) method. Moreover, with the help of an accurate FE model, a database can be simulated for an arbitrary number of load cases, damage types, and uncertainties at minimum effort [18]. In the technique presented in [19], experimental measurements of the actual structure modify the physical parameters of an FE numerical model. This type of optimal FE model developed on the actual structures can also be used to simulate damage status. Similarly, in article [20], the FE model was updated with data collected from the sensors fitted to the vertical truss bridge. Wavelet-based damage detection criterion is also used to assess the impact of a vessel on bridges. Furthermore, by using an FE model, a large amount of labeled data, which are influential but not possible to simulate in real conditions, can be simulated for machine learning-based SHM. In this research work, a 3D computer model is developed using CAD software, considering the composition of materials and their mechanical characteristics. Research is further extended to create an FE analysis mesh. A library of training data is generated by performing simulations for different loading conditions using the FE model.

Optimal Placement of Sensors (OPS)

Sensor array is a fundamental part of the SHM system, as it acquires real data from structures. The quality of data directly depends on the optimal placement of sensors (OPS). The two main requirements of OPS are to minimize the number of sensors and to maximize the accuracy of the data. Therefore, choosing the right strategy of OPS is critical to any SHM system. The review article [21] discusses current work in the field of OPS for monitoring schemes based on vibration, strain, and elastic wave. It also highlights different optimization algorithms with their respective benefits and limitations. Since physical testing of various sensor deployment schemes is not feasible, a suitable approach is to use a computational model of structures to determine the optimal sensor configuration. Article [22] provides a comprehensive review of computational methodologies for OPS in SHM and summarizes various algorithms with an emphasis towards evolutionary algorithms and its variants. Additionally, it discusses evaluation criteria and the appropriateness of computational methods for specific SHM applications. In [23], sensor placement is completed by using a hybrid optimization algorithm based on FE grids and sensor distribution index. This approach provides reasonable FE grids for optimal placement of sensors that overcome redundancy of information. Article [24] proposes genetic algorithms for optimal sensor placement and considers a redundancy elimination FE model designed based on a sub-clustering strategy.

Sensor Technologies

In the last two decades, there has been a rapid development in sensing technologies. As such, the use of machine vision-based technology in the field of SHM has increased. A comprehensive review of the vision-based methods and applications in SHM is presented in [25]. The review article concludes that machine vision-based technology is widely used to measure 2D and 3D structural displacement, strain, and other additional parameters. Moreover, it can be used to conduct structural parameter identification and damage analysis. Hence, vision-based methods integrated with other sensing techniques have the potential to provide more valuable information of SHM systems. The efficiency of integrated infrared imaging in detecting damage to bridges is summarized in [12]. The article explains the computer vision-based techniques focusing on contactless vibration monitoring and load quantization. The vision-based SHM system proposed in [26] uses a camera as the sensing element to extract displacement and strain data of structures. Furthermore, a comprehensive analysis of the lab structural test has shown a good agreement between camera-based structural responses and the damage detection recorded by conventional contact sensors. Structures like highway bridges are validated with this approach. In this research, small infrared cameras integrated with surface markers are used to track displacements and remotely record bridge deformations.

Sensor Placement

SHM techniques are more widely used due to the lower deployment cost. Moreover, Optimization of Sensor Placement or OSP is the best choice to reduce the cost of an SHM system without compromising the quality of the monitoring approach. This article is aimed towards researchers working on OSP as well as practicing engineers in the field of SHM [21]. It covers three techniques that are most commonly used and accepted in the SHM community: vibration-based monitoring, strain monitoring, and elastic wave-based monitoring. Keeping in mind the structural and execution demands, the multi-objective optimization or the problem definition is also discussed. Different optimization algorithms have been implemented in this study. Furthermore, researchers have highlighted various pitfalls and their appropriate countermeasures to overcome the shortcomings of SHM.

In this research, the global multi-objective optimization of sensor locations for structural health monitoring systems is studied [27]. Using the Finite Element Method (FEM), a laminated composite plate is demonstrated and placed into the modal analysis. To search for the optimal locations of sensors, multi-objective genetic algorithms (GAs) are adopted. In structural dynamics, numerical issues rising in the selection of the optimal sensor formation are discussed. Using the composed information by Fisher Information Matrix (FIM) and mode shape interpolation, a method of multi-objective sensor locations optimization is presented in this research.

The main contribution of this research is to tackle the OSP problem by establishing a new performance metric that is rooted in Bayes' risk formulation. To maximize the likelihood of detection, or reduce the false alarm rate with minimal overall cost, this performance metric is specially designed for and directly addressed to the SHM objective. Thus, this technique can be considered a useful tool for designing SHM sensor arrays. It is focused on active sensing and producing an appropriate statistical model of the wave propagation and feature extraction process using guided ultrasonic waves.

Research in [29] discusses monitoring a self-governing system in which a Wireless Sensor Network (WSN) is used. Through the placement of a set of backup sensors, the WSN objectives attain a fault tolerance that results in accuracy in its measurements. This research follows a distributed method in which the nodes are grouped into clusters. The separate point, remote point, and critical middle point are determined from each cluster where the backup sensors are located. The energy is consumed, and the lifetime of the sensor nodes is greater than before through the fault tolerance mechanism. The validation demonstrates the benefits of using the fault tolerance mechanism.

For optimal sensor placement and damage identification, this article proposes an efficient methodology in laminated composite structures [30]. To develop a reduced-order model for Optimal Sensor Placement (OSP), this method first applied a model reduction technique that is iterated in the improved reduced system (IIRS) method. By formulating and resolving an optimization problem for finding the best sensor locations, the OSP strategy uses the Java algorithm. In order to identify and measure any stiffness reduction induced by the damage, this approach uses the measured partial modal data from optimized sensor locations. For this implementation, specifically where the damage number of elements and the modal flexibility transformation are taken as the constant design variables and the objective function, the damage identification problem is expressed as an optimization problem. To solve the optimization problem for determining the definite damage locations and levels, the Jaya algorithm is implemented again. To prove the feasibility and efficiency of the proposed method, numerical simulations of a three cross-ply (0°/90°/0°) beam and a four-layer (0°/90°/90°/0°) laminated composite plate are carried out.

Research in [31] an overview of current, state-of-the-art technologies in Sensor Placement Optimization for SHM problems. There is a great deal of progress in some important areas, including in methods that stimulate robustness and in modeling ambiguity when dealing with sensor placement optimizations, beginning with effects within measured data and failures within the sensor network. The central focus of this study is to highlight emerging trends particular to SHM system development.

This study focuses on the construction of precise strain maps for large-scale structural components and the development of optimal sensor placement within a hybrid dense sensor network [32]. In large-scale structures, understanding precise strain-maps is imperative for better strain-based fault analysis and diagnosis health management. To reduce type I and II errors, and an adaptive mutation-based genetic algorithm, this study creates a unique and precise objective function. The objective function authorizes sensor placement and is based on the linear combination method while increasing information entropy. By applying a genetic algorithm that influences the concept that not all potential sensor locations hold the same level of information, OSP is achieved. The experimental analysis demonstrates the ability of the learning gene pool to efficiently and frequently discover a Pareto-optimal solution quicker than its non-adaptive gene pool equivalent.

This research presents the advancement of SHM technology that has been implemented in long-span bridges [33]. The techniques of modal identification, signal processing, and damage identification, including data analysis and condition assessment, were reviewed. To advance the understanding of the utilization and examination of an SHM system for long-span arch bridges, an SHM system of a long-span arch bridge (the Jiubao Bridge in China) was thoroughly integrated. Additionally, potential future trends and challenges of this system were outlined.

In a structural health monitoring system, data is obtained from sensors for a reliability evaluation of the structure, and a false alarm will often be generated if a faulty sensor is present [34]. In this study, in order to identify a sensor fault, a technique based on the generalized possibility ratio and correlation coefficient is presented. By applying a minimum mean-squares-error algorithm under the operational condition, and through an evaluation of each sensor in the sensor network, the acceleration response of a bridge is assumed to be Gaussian distributed. Between the estimation and measured data, the classification features five common sensor fault types which are considered, with two correlation coefficients calculated. To categorize the type of sensor fault, a disturbed binary tree method is implemented. Numerical and experimental analysis show that the proposed technique is robust in the detection and classification of sensor faults.

In this research, the design of an experimental system with a reduced number of sensors for the structural health monitoring of the historical bridge of Posadas (Córdoba, Spain), designed by the eminent engineer Eduardo Torroja in 1957, is presented [35]. It is necessary to rely on a sufficiently precise numerical model, as most OSP techniques are model-based. With a large number of accelerometers, a wide vibration-based functioning modal investigation is conducted. Using a genetic optimization algorithm, a three-dimensional FEM of Torroja's bridge is restructured based on the experimentally identified dynamic properties. To design an experimental setup with a limited number of sensors, the OSP approach is applied for long-term observing commitments. The experimental validation proves that some sensors are used to precisely measure the main resonant occurrences and mode shapes.

Another interesting case is the presentation of a unique approach for piezoelectric (PZT) wafer-network placement that was implemented using a genetic algorithm (GA) [36]. While using the smallest possible number of sensors, the proposed objective function maximizes the exposure of the observed area presented by a set of control points. Simulation results are presented for three cases, a square panel with geometrical discontinuity, a T-shaped panel, and a cargo door of an Airbus A330-200 airplane. The difference between the primary and the improved experimental results indicates a major improvement in the coverage level. Using ultrasonic excitations at different frequencies, experimental verification was performed on the square panel and a part of the cargo door. Artificial costs were identified and then restricted with an error rate of not more than 4% of the highest distance in the geometry.

This research discusses the inclusive analysis of computational methodologies for OSP in SHM [37]. By using evaluation criteria for sensor configurations, the problem formulation of OSP is presented. Evolutionary algorithms and their improved variants are discussed in detail for the existing optimization approaches for sensor placement. For sensor configuration determination, this research highlights the most commonly applied criteria and optimization approaches.

This research presents an approach to the optimal placement optimization of sensors' locations for SHM [38]. The finite element method (FEM) was the key component in the structuring of the system. To find the best sensor distribution and to cover a precise number of low-frequency modes, genetic algorithms (GAs) are then implemented. The performance of sensor delivery methods is examined by numerical results.

A huge amount of raw data and processed data are some of the concerns in SHM. A powerful data management tool, which obtains, sorts, stores, shares, and recalls data, as well as delivers a digital environment, is Building Information Modelling (BIM) [39]. The primary purpose of this study is to use BIM to observe the data of monitoring systems, especially the SHM system. To validate the probability of generating and visualizing information about the sensors installed in the structure for SHM, a four-story office building is displayed in Revit architecture. To manage the sensor data in real time and to ensure the sensor's information is up-to-date, the BIM model is made dynamic by connecting appropriate external resources associated with the sensors.

Machine Learning Algorithms

In recent years, significant developments in the field of machine learning and artificial intelligence have made machine learning-based SHM an extensive research topic. A comprehensive review of machine learning algorithms applied in civil SHM is presented in [40]. This discusses the efficacy of deploying machine learning algorithms in SHM. In

[41], operational modal analysis is integrated with artificial neural networks (ANN) to detect damages in structures. Using natural frequencies and mode shapes of structures, a simple numerical model database is built with varying stiffness. The ANN with 1,400 neurons and one hidden layer is fed with the inputs of model properties and the outputs of stiffness reductions. The neural network model detects the damage of the structure in terms of damage locations and levels. This monitoring strategy is successful in detecting single column damages to a building without any errors.

Machine learning algorithms are generally classified as either supervised or unsupervised learning. In [42], a comparison is made among three supervised machine learning algorithms: namely, k-nearest neighbor (KNN), support vector machine (SVM), and random forest classifier (RFC), in order to predict the structural damage to concrete structures. Algorithms were tested on publicly-available test results with varying stiffness and mass conditions. Results show that RFC outperforms the other two machine learning techniques. Multivariate data-driven and machine learning approaches were used for the detection and classification of damages [43]. The reduced dimension data processed using Principal Component Analysis (PCA) and other pre-processing techniques was used to train the networks: namely, Subspace KNN, Bagged Trees, Weighted KNN, Fine KNN, Coarse KNN, Subspace Discriminant, and Boosted Trees. The first four methods exhibit better accuracy compared to other methods. In [44], a machine learning approach was applied to noisy data from low-cost accelerometer sensors to identify the status of structures. Convolutional neural networks (CNN) outperformed SVM and KNN, even with noisy data used as inputs. A detailed review of machine learning algorithms in bridge health monitoring is presented in [45]. Addtionally, the article presents the advantages, weaknesses, and applications of different algorithms when applied to bridge health monitoring. In [22], auto-detect anomaly in SHM incorporating computer vision and machine learning methods are employed. Deep neural network (DNN) machine learning is used for training and auto-detecting anomalies in the image vectors obtained from time series acceleration data of an actual long-span bridge in China. A DNN-based bridge health monitor is proposed in [46]. The mid-span temperature and stress of the bridge were used as input parameters to train the network. In [47], a novel method is proposed to predict dynamically reconstructed responses using DNN. The dense network is trained on available acceleration responses from Guangzhou New Television Tower in China, and the reconstructed data performed well, both in time and frequency domains. Drawbacks related to feature extraction, a reduction in the number of parameters, and gradient and noise immunity can be overcome using dense neural networks.

Since 2014, ensemble machine learning-based SHM has gained momentum. An ensemble machine learning-based SHM is presented in [48] to classify the failure mode and load-bearing capacity of reinforced concrete. Boosting algorithms are employed for both classification and regression. Furthermore, boosting algorithms have shown better performance than single learning algorithms. The ensemble learning approach has been employed for crack detection using deep convolutional networks as well [49]. The data set consists of images of the bridge towers and anchor chambers of a suspension bridge. In [50], a gradient boosting-based machine learning algorithm is considered to predict the damage of reinforced concrete panels under impact bearing. The model is tested on the impact bearing reinforced concrete panels by using experimental data. A reasonable accuracy level of around 75% is achieved, promising a new and effective approach that can be applied to other complex structures as well. A summary of the discussed machine learning-based SHM is given in Table 1.

Table 1 Summary of Machine Learning-Based SHM

References	Machine Learning Technique	Structure	Input	Outcome
Smarsly et al., 2016	KNN, SVM, RFC	Publicly available test results	Stiffness and mass conditions	RFC gave better prediction
Vitola et al, 2016	KNN variants,	Aluminum plates in different actuator phases	Signals captured using piezoelectric sensors	Damage detection and classification
	Subspace discriminant, bagged trees			
Ibrahim et al., 2019	KNN, SVM, and CNN	Multi-floor buildings	Accelerometer traces	CNN gave better results
Bao et al., 2019	DNN	Actual long bridge	Acceleration data	Auto-detect structural anomalies
Chen et al., 2019	DNN	Bridge	Stress and temperature	Bridge health diagnosis
Feng et al., 2020	Ensemble learning (boosting algorithm: gradient and adaboost)	Reinforced concrete	Geometric dimensions, material properties	Failure mode classification and load bearing capacity prediction
Kailkhura et al., 2020	Ensemble learning using DCNN	Bridges	Images of bridges	Crack detection
Thai et al., 2019	Gradient boosting	Reinforced concrete panels	Boundary conditions and others	Failure modes

Chapter 3 Systems Engineering Framework

Overview

The formulation phase for any project is updated as required throughout the project's life cycle. This chapter provides the specifics of the technical effort. It describes the technical processes used, the application of processes by using appropriate activities, the organization of the project to accomplish the activities, the information flow within the system, the decision-making structure, and the resources required for the accomplishment of activities. Critical events drive activities during any phase of a life cycle (including operations) and are the basis for the integration of the processes. This chapter presents the communication bridge between the project management team and the engineering discipline teams. It also facilitates effective communication within the discipline teams. Furthermore, this document provides a framework to realize the appropriate work products that meet the entry and exit criteria of the applicable project life-cycle phases to provide necessary information to management in order to assess technical progress.

As this chapter showcases the results of studies performed on a smaller scale, it reflects the effectiveness of the design proposals. This chapter shall also highlight the required budget, effectiveness, and schedule. The presented information is useful to project stakeholders, such as project developers, researchers, department, faculty, and the university. The primary audience is the university's project-funding facility, which assists the researchers in implementing useful findings in real-world applications.

Rationale

The problem background that is being solved by the research is identifying the design and materials suitable to make an infrastructure design that would result in the least amount of failures. A study identifying the requirements of a specific structural design suitable for certain conditions was missing previously and is therefore being addressed in this research. Results can be foreseen digitally by incorporating engineering and machine learning simulations, and their success score can be pre-calculated. Questions posed by previous studies remain relevant and need to be addressed by engineers, which is what this research aims to accomplish. This experimental research presents computer software and simulation based on machine learning prediction models with a high accuracy rate of 76%. Machine learning has enabled the prediction of high performing materials and infrastructural designs.

Systems Engineering Framework

Systems engineering ensures significant development and delivery of capabilities by using a set of integrated, disciplined, and consistent analytic and technical management processes throughout the infrastructure lifecycle. While systems engineering touches many of the other processes across the development lifecycle, this research dwells on the application of agile principles that allow for creativity. The twenty-first century provides an exciting opportunity for systems engineering. Indeed, increasing technological complexity results in new challenges in architecture, networks, hardware and software engineering, and human systems integration. This research study on SHM of civil infrastructures, such as bridges, deals with technological complexity by leveraging the sensors and techniques (drones and cameras) emerging from the Internet of Things (IoT) revolution that continually incorporates new technology. Adapting these new technologies to artificial intelligence or machine learning algorithms for data analysis reveals insights into the health of civil structures that were much more difficult to ascertain earlier, even with significant investment and long downtime of infrastructure.

The ISO/IEC/IEEE 15288-2015 systems engineering processes were refined to reflect agile principles at the agile Working Group held during the International Conference on Systems Engineering (INCOSE). In an agile environment, systems engineering requires tailored methods and processes to deliver incremental capabilities. Therefore, it demands a disciplined approach for coordinating parallel requirements, elaboration and prioritization of technical developments, operations, and sustainment activities. Systems engineers play an essential role in operational, technical, and programmatic integration, as expressed in the core agile software development tenet of active collaboration among developers, users, and other stakeholders. Program leaders must encourage systems engineers to engage with developers, testers, users, and other stakeholders in their disciplined engineering processes.

Agile development requires proactive collaboration among enterprise architectures, platform architectures, and related development efforts, where each stakeholder group shares concerns and opportunities regarding the successful release and system delivery.

This will enable smaller yet faster capability deliveries to consistently track the accurate SHM of civil structures.

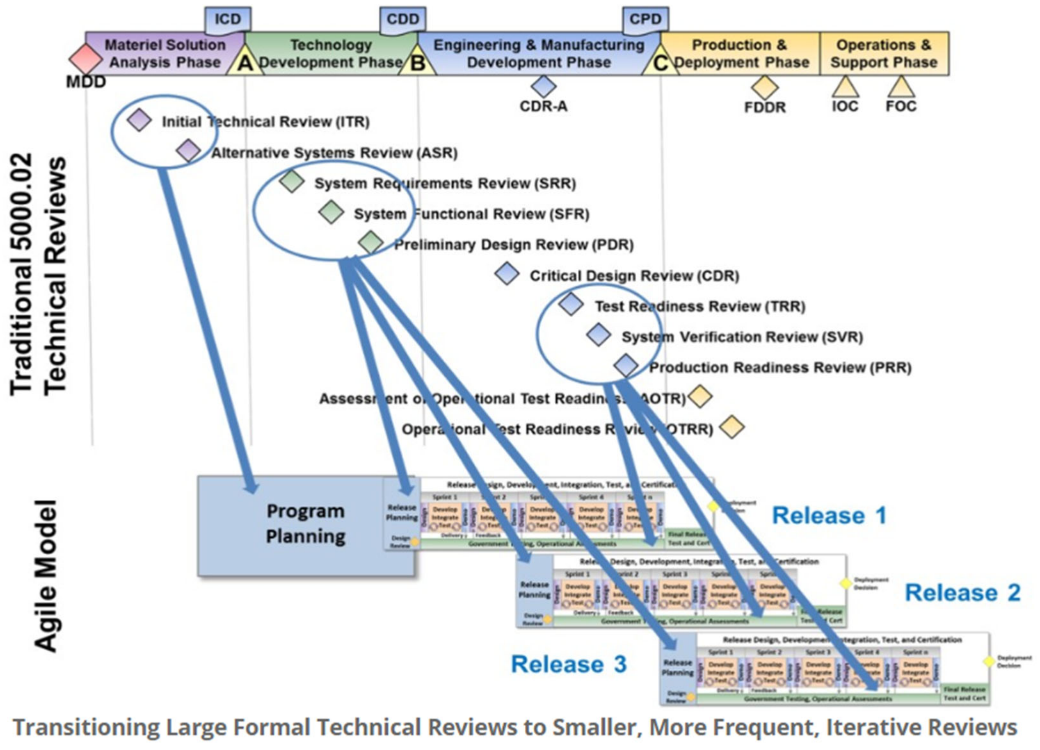

ICD
CDD
CPD
Materiel Solution Analysis Phase
A
Technology Development Phase
B
Engineering & Manufacturing Development Phase
C
Production & Deployment Phase
Operations & Support Phase
MDD
CDR-A
FDDR
IOC
FOC
Traditional 5000.02 Technical Reviews
Initial Technical Review (ITR)
Alternative Systems Review (ASR)
System Requirements Review (SRR)
System Functional Review (SFR)
Preliminary Design Review (PDR)
Critical Design Review (CDR)
Test Readiness Review (TRR)
System Verification Review (SVR)
Production Readiness Review (PRR)
Assessment of Operational Test Readiness (AOTR)
Operational Test Readiness Review (OTRR)
Agile Model
Program Planning
Release 1
Release 2
Release 3
Transitioning Large Formal Technical Reviews to Smaller, More Frequent, Iterative Reviews

Figure 1 Systems Engineering Framework

In the left-hand side of Figure 1, traditional systems engineering practices are consolidated and combined into an incremental model. This approach provides more timely data on the structural health of bridges and other civil structures that might otherwise deteriorate unknowingly, potentially resulting in catastrophic failure.

The danger and inconvenience caused to the public due to the closure of bridges are averted using agile techniques. The principle is very similar to the combination of development and operational (DevOps) software development models that can be developed and released, providing quicker access to users. In the case of agile systems engineering, knowledge of the SHM of civil structures and the detection of potential problems are delivered using predictive maintenance analysis. The lifespan of the civil structure increases and the usability of the structure benefits the public. Engineers should conduct continuous interdisciplinary systems engineering reviews to find the right balance between structure and flexibility in order to deliver usable capability aligned with the needs of users. These changes to the systems engineering model, from traditional to agile, lead to the systems engineering V-model that can be tailored to the needs of the developer in scheduling the SHM activities.

Systems Engineering Model

The best-suited systems engineering model is the V-model that considers the procedures in a step-by-step manner, mitigating the chances of errors in each step, and subsequently ensuring stepwise success (Figure 2). The V-model guides the chaptering and realization of the proposed project. Through it, the following objectives are intended to be achieved:

- Minimization of project risks.
- Improvement and guarantee of quality.
- Reduction of total cost over the entire project and system life cycle.
- Improvement of communication between stakeholders.

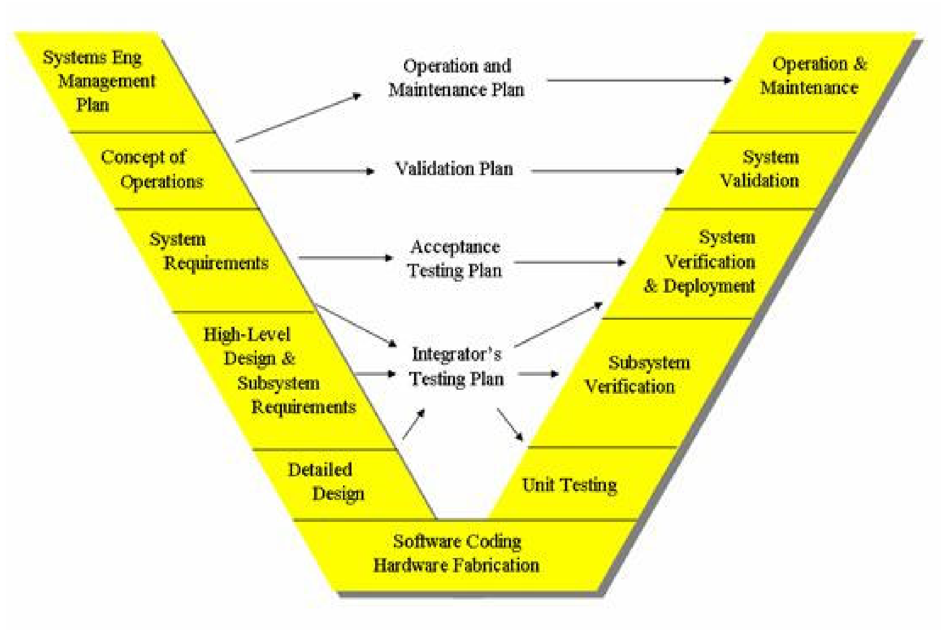

Systems Eng Management Plan
Concept of Operations
System Requirements
High-Level Design & Subsystem Requirements
Detailed Design
Software Coding Hardware Fabrication
Unit Testing
Subsystem Verification
System Verification & Deployment
System Validation
Operation & Maintenance
Operation and Maintenance Plan
Validation Plan
Acceptance Testing Plan
Integrator's Testing Plan

Figure 2 Systems Engineering V-Model

The final system operates by using remote sensing technology to identify the measurement of deformation at multiple points. The system includes:

- An algorithm for optimal sensor placement.
- A modeling framework for SHM, used to create the training datasets.
- A library of known structural behaviors from the available structural model.
- A machine learning algorithm to predict and correlate data sets to the known failure modes.

A good visible representation of these processes is shown in a closed feedback loop that ensures any change made to these technical processes is reflected throughout the entire design of a civil structure (Figure 3). In this representation, the technical process is made compact and reduced into five essential functions.

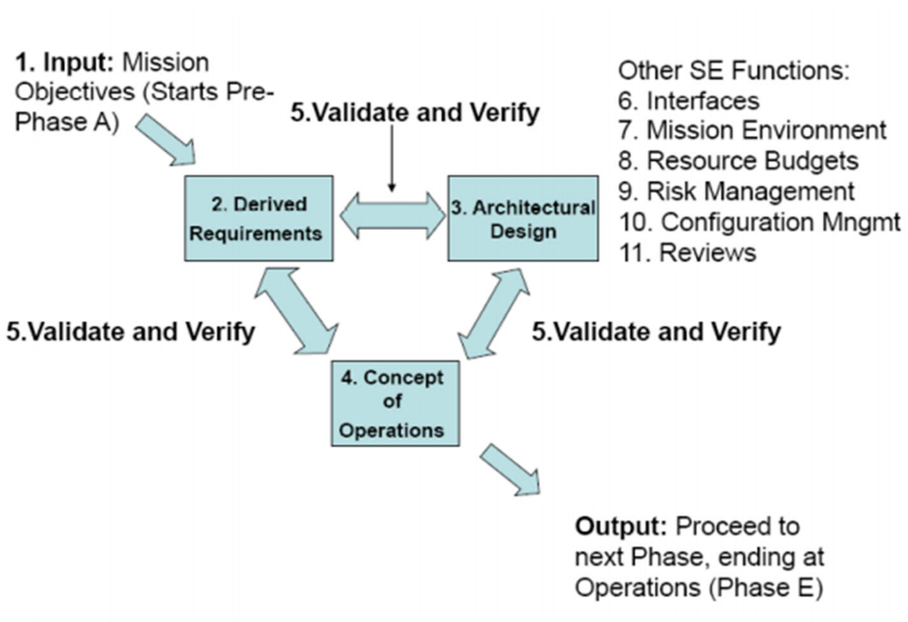


Figure 3 Systems Engineering Functions

Analysis of Requirements

The requirements to run this project successfully relied mainly on the application and accuracy of a machine learning algorithm. The main advantage of using a machine learning algorithm is that it is self-correcting. Thus, a massive computational power is required for its operation. A better computational power enables better results [14]. This will ensure the correct prediction of failures in materials and designs, thereby resulting in higher rates of accident prevention [51] .

Requirements are listed as follows.

- Computational power
- Powerful workstations

- Data sets (testing, modeling, and actual data)
- Materials suitable to test small-scale bridges/infrastructure

Stakeholder Analysis

It is essential to perform a stakeholder analysis for conducting the proposed research study and implementation in order to highlight the benefits each stakeholder receives. Every stakeholder invests time, energy, and overall interest to obtain the best results through a successful design and analysis. Moreover, stakeholders have a certain degree of power over the processes. This presents a unique power-interest grid for every stakeholder [8]. Additionally, stakeholders often have to communicate in order to drive the project forward. In this research project, the academic stakeholders are researchers and the academic organization, while the practical stakeholders are engineering firms, investors, and supply chain managers of the construction materials.

SWOT Analysis

SWOT analysis is a critical analysis to be performed on projects related to machine learning models. It is a good practice to highlight the strengths, weaknesses, opportunities, and threats faced by the proposed project [9]. The SWOT analysis of this research project is presented in Table 2.

Table 2: SWOT Analysis

Strengths	**Weaknesses**
Ability of machine learning computations to self-correct their results. Effectiveness of the end product. Save on material costs by using simulation and computation. Minimal manual work.	Dependency on high computational power. Dependency on accurate data sets to make effective models. Testing overheads. An on-site AI practitioner is required for effective monitoring.
Opportunities	**Threats**
The idea can easily be expanded to suit other industries, such as textile manufacturing. Scalability is easy. Novel idea will attract sponsors. No added complexity.	Accuracy decreases with multiple iterations. The application of the same advanced algorithms. by competitors, should it get late to implement.

Risk Analysis

It is essential to conduct risk analysis when funding or capital costs are involved in order to mitigate risks by correctly analyzing them beforehand [52]. A significant risk is the ineffectiveness of end products, resulting in the wastage of materials. This can be subsumed into more commonly identified forms of risk found in all machine learning projects, namely, unexpected behavior and unintended consequences. The use of unverified technologies, errors in monitoring, and other similar factors increase the risk of emergencies, the consequences of which can lead to human casualties. Moreover, accidents and disasters cause not only severe moral and social shocks, but also financial losses that occur due to the inoperability of the object. For this reason, NN can become an indispensable tool for SHM. It is a compelling modeling method that allows the reproduction of too complex dependencies, particularly the ones that are non-linear. At the same time, neural networks learn from examples; thus, they can be a good choice. Nevertheless, they cannot completely replace existing methods or the work of the specialists, only complement them.

Research also addresses the development of various sensors and prediction systems that facilitate focusing on the resources in high-risk areas of failure as well as the structures most prone to immediate failures according to various reports from American Society of Civil Engineers. Current studies regarding the inspection and calculation of structural health data via sensors have focused almost exclusively on the application of fiber optic sensor systems [6]. Fiber optic sensors are highly reliable and embedded into large concrete structures during the construction phase [7]. Some of their identified risks are given in Table 3. To determine the potential risks, SHM will be evaluated using the checklists found in [53]. These checklists help to identify potential risks in a generic sense. The project will then be analyzed to determine any project-specific risks.

Product Size Risks

- Estimated size in lines of code (LOC): Structural health system (SHS) will have a code with about 10,000 lines.
- Degree of confidence in the estimated size: The confidence in the estimated size is very high.
- Deviation from the average of previous products as a percentage: A deviation of 20% from the average is allowed.
- Multiple users: The number of users will be relatively low. There will be one user per instance of running the software, as the software was not planned for multiple users.
- Number of projected changes to the requirements: Three possible projected changes to the requirements were estimated. The changes will occur when the requirements identified in the initial stage are not required at implementation. It may happen when the customers' requirements are verified by interacting with them.
- Amount of software reusage: Reusing is very important to get the project started. The CAD methodology is fairly straightforward to reuse. Previous programs used to code with CAD will be reviewed, and relevant codes will be extracted.

Business Impact Risks

- Effect on company revenue: None. SHS will be distributed as a highly cost-effective software. It will be a hot-selling software that will save massive costs in the construction and maintenance of bridges. As it will be developed by using some pre-existing and open source tools, no business risks are involved.
- Visibility of product to senior management: To achieve this, plots of safety factor vs. node, and deformation vs. node, are generated separately. They represent the complete output of the model under a specified loading condition (e.g., scenario P12500 represents a load of 2,500 lbs. in the predetermined position 1).
- Reasonableness of delivery by deadline: Fairly reasonable. The project deadline was established before the project was begun. The initial chaptering for SHS was executed with the deadline in mind. The scope of the project was limited to keep the project "doable" within the allowed period.
- Number of other systems/products that the proposed product must be interoperable with:
    - CAD Engine, which is included with Python.
    - Machine Learning algorithm.
    - Sensor.
- Amount and quality of documentation that must be produced and delivered to the customer: The customer will be supplied with a complete online FAQ and help tool as well as a user manual for SHS. The customer will have access to all of the development documents for SHS, as the customer will also grade the project.
- Governmental constraints in the construction of the product: No relevant constraints are known.
- Costs associated with late delivery: Late delivery will prevent the customer from issuing a letter of acceptance for the product, which will result in an incomplete grade.

Customer-Related Risks

- Have you worked with the customer in the past? No.
- Does the customer have a solid idea of what is required? Yes, the customer has access to both the system requirement specifications and the software requirement specifications for the SHM project.
- Is the customer willing to establish rapid communication links with the developer? Yes, the customer can access all project developers through e-mail and in-person.
- Is the customer willing to participate in reviews? Unknown. While the customer will likely participate if asked, no inquiry has been made at this time.
- Is the customer willing to let your people do their job? Yes. As the SHM project is a senior design project, the customer is available if needed but does not interfere with development operations.

Technological Risks

- Is the technology to be built new to your organization? SHS is a software tool to be used in SHM. Development team members are familiar with its development, as well as the necessary dataset implementation.
- Do the customer's requirements demand the creation of new algorithms or input or output technology? No. SHS will be implemented using existing algorithms. Input and output will be handled traditionally.
- Do requirements demand the use of new analysis, design, or testing methods? No. The development team will implement existing analysis, design, and testing methods for this project.
- Do requirements demand the use of unconventional software development methods? No. SHS uses Python code in header files which is conventional. It is also integrated with CAD, which is conventional as well.
- Is the customer uncertain that the functionality required is "doable"? No. The customer has full confidence in the project described in the system specification document and the software specification document.

Table 3: Risk Assessment

Risks	Category	Probability	Impact
Computer Crash	TI	70%	1
Late Delivery	BU	30%	1
Lack of Development Experience	TI	5%	2
Lack of Available Dataset	TI	40%	2
Poor Quality Documentation	BU	35%	2
Deviation from Software Engineering Standards	PI	10%	3
Poor Comments in Code	TI	20%	4
Equipment Failure	TI	70%	1
Technology Not Meeting Expectations	TE	25%	1
End Users Resisting System	BU	20%	1
Changes in Requirements	PS	20%	2
Less Reuse than Expected	PS	60%	3

Impact Values:

1 – Catastrophic
2 – Critical
3 – Marginal
4 – Negligible

In most civil infrastructure contracts, many of the systems engineering processes are executed by the contractor(s). As these processes overlap with the responsibilities of the government in some cases, it is essential to understand whether these activities are accomplished effectively. The systems engineering technical processes that need to be monitored include the following:

Design definition: The overall system design will and should evolve. All stakeholders must be involved in design decisions.

System Analysis: This is an ongoing process to ensure that incremental development of the solution remains stable, coherent, and aligned with the needs of all stakeholders.

Verification and Validation: The release process must continuously verify the built works (verification) and meet the needs (validation), which will satisfy the users.

Transition: This refers to moving incremental solutions into operations, which must be accomplished with care to avoid update fatigue on the users' side while allowing the solution to evolve.

Operation: In an agile environment, development continues while initial releases are being used in actual operations. It is essential to monitor and measure the performance of the delivered increments and to have a mechanism in place to respond to "real user" feedback.

Maintenance: Fielded systems may break. Maintenance activities should be fed into the product backlog and prioritized accordingly. Subsequent releases are technical upgrades to the existing system and should be managed through users' maintenance processes, too.

Disposal: As a new functionality is added to a fielded and incrementally developed system, legacy systems may eventually need to be removed from service. Sometimes, early functionality of the new system may need to be removed when replaced by an improved functionality or when they are no longer needed.

Quality Assurance: Going beyond tests, maintaining acceptable software engineering practices are essential to preserve the portability of codes and allow future unanticipated changes to the fielded system.

Decision Analysis

The five technologies mentioned below have been reviewed.

1. Passive Electromagnetic RFID Sensors for crack detection [48] [47].
2. Computer Vision: Kinematic SAMI, a new real-time multi-sensor data assimilation strategy that uses high-speed and high-resolution cameras [46].
3. Liquid Level Sensing Systems (LLSS) as an add-on to the bridge structure to measure deflection and deformation [49].
4. Ultrasonic Sound Wave Technology [50].
5. Infrared Thermography, UAV Photogrammetry, and GPR [54] [55].

Decision analysis for agile occurs at critical points in development, specifically when issues arise. The development team must make decisions about development steps and perform designs based on a clear understanding of risks and options. This can be challenging, since an optimum design and implementation solution are typically unknown. Additionally, in many cases, the risks and options are unknowable due to the inherent complexity at the outset of a program. When this is the case, the team should resist the urge to over-engineer or design an optimal solution in advance. Instead, the team should proceed based on an agreed-upon preferred direction and then assess progress and course-correcting based on an analysis of specific experiences. Typical agile decisions center on adjusting the grouping of user stories, determining which sprints are released to users, resizing and splitting the design, implementing separate stories, and assessing the impact of removing requirements from the backlog.

In this research, SWOT analysis was used due to its simplicity and straightforward path towards make a decision. It is also well-suited to the agile methodology which is employed in this study. Hence, five potential contactless sensor-based methods were studied:

- Passive RFID sensors have the drawback of signal interference due to surrounding obstructions. The transmitting power of the RFID sensor is severely limited due to no battery being mounted on the sensor.
- The LLSS structure requires a costly development phase. It is subject to damages due to weather conditions, such as wind, and various abnormal loading conditions.
- The Ultrasound technology was sensitive to both environmental conditions and its proximity to the bridge.
- Computer vision and infrared methods were both drone-mounted camera options that could be operated remotely from a safe distance with visible evidence being revealed in real time and at scale (Figure 4 and Figure 5).

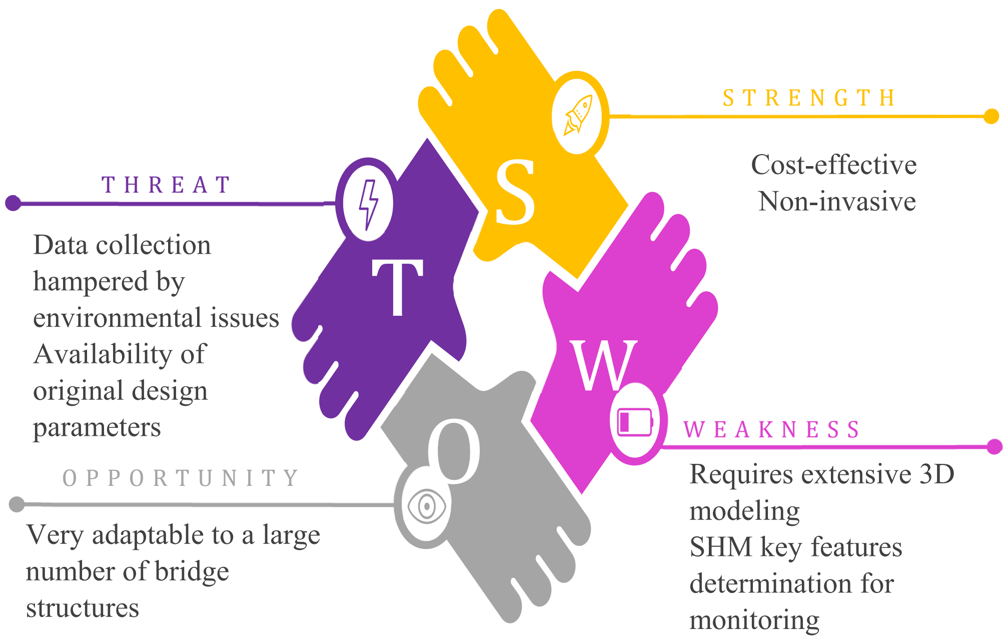

STRENGTH
Cost-effective
Non-invasive
THREAT
Data collection hampered by environmental issues
Availability of original design parameters
S
T
W
O
WEAKNESS
Requires extensive 3D modeling
SHM key features determination for monitoring
OPPORTUNITY
Very adaptable to a large number of bridge structures

Figure 4 SWOT Analysis of Computer Vision

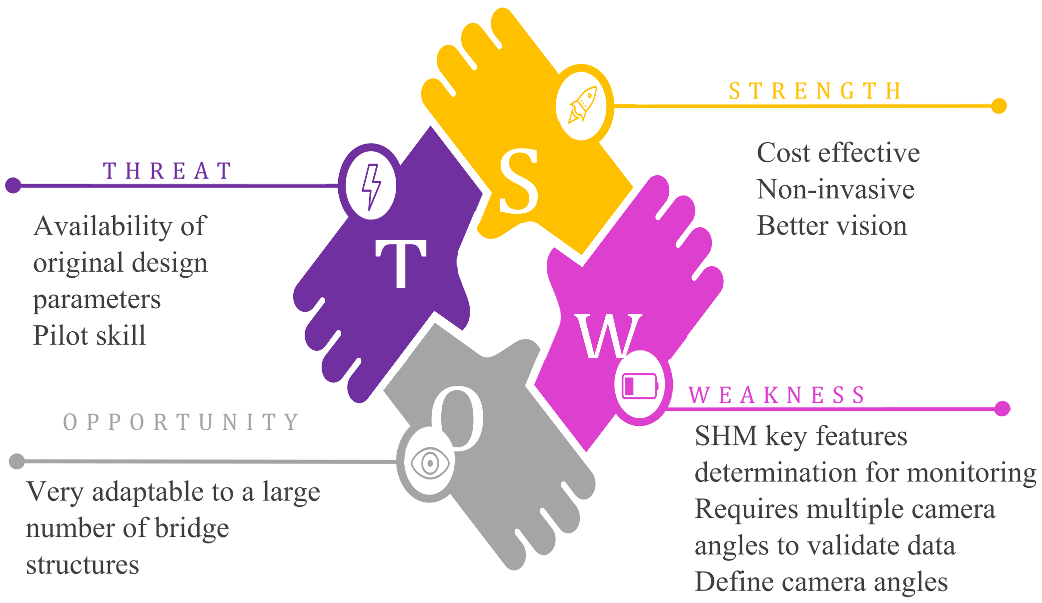

STRENGTH
Cost effective
Non-invasive
Better vision
THREAT
Availability of original design parameters
Pilot skill
S
T
W
O
WEAKNESS
SHM key features determination for monitoring
Requires multiple camera angles to validate data
Define camera angles
OPPORTUNITY
Very adaptable to a large number of bridge structures

Figure 5 SWOT Analysis of Optitrak

Chapter 4 Concept of Operations

As per the literature, use of non-contact camera sensors in SHM has added advantages compared to traditional contact sensors. However, the majority of previous work focuses on the optimal placement of contact sensors for SHM. To fill this gap, the present work proposes a novel filtering algorithm for optimal placement of contactless sensors based on strain, deformation, node location, and the safety factor. Furthermore, it is evident from the above-discussed literature that machine learning-based SHM is preferable to other SHM systems. Moreover, past works have shown excellent results with gradient tree-based machine learning algorithms for structural monitoring of reinforced concrete panels. With an extensive literature survey, it was found that further adaptation of NN is required in order to increase accuracy. It is clear from the existing literature that there is a need for further study of the application of NN and prediction ability when training is based on computer-simulated data. This becomes extremely valuable when live data acquisition is feasible and/or destructive tests are not possible. There is a gap in the literature concerning experiments that measure prediction capability of artificially trained NN on real-life structural behavior. Additionally, the use of gradient boosting algorithms for structural monitoring shows great promise but lacks sufficient attention. Consequently, in this article, a gradient boosting neural network is considered as a suitable machine-learning algorithm for bridge monitoring and damage prediction. The proposed SHM system integrates the filtering algorithm for active failure detection, simulated training data, and known failure cases with the gradient boosting machine learning model for bridges, then conducts trained with simulated data and validates using input from a physical model.

The following failure modes have been identified from previous research or through extrapolation of material properties under expected lifetime loads. The proposed research methodology includes identification of suitable techniques to record structural health and structural behavior data, development of a structural modeling framework to simulate multiple structural loads and failure modes, and analysis of sensor data into a reliable prognosis system.

- Loading Fatigue [41], [47], [48], [15].
- Vibration Fatigue [41], [47], [48] [15].
- Mechanical Overload [46], [47].
- Creep (Moaveni et al., 2013), [46].
- Thermal Shock [41].
- Corrosion [41], [47].

Solution Approach

The solution approach, including a specification of techniques used, is described as follows (Figure 6). Firstly, the structure is scanned, and a 3D CAD based model is created (1), which is then continued to an FE analysis model. An analytical approach is followed to simulate loading conditions (2) and a library of training data is generated (3). The data set is later fed to a machine learning algorithm (4) that has two more types of data as inputs: other known failure cases (5) and data generated by the sensor array (6) of the real-life model. The NN algorithm then develops a correlation between the different datasets. It determines the percentage of correlation between the sensor input and the known scenarios (training or surveyed) and produces a prediction based on this correlation (7). The detailed procedure followed for a test case is explained in the applied methodology section.

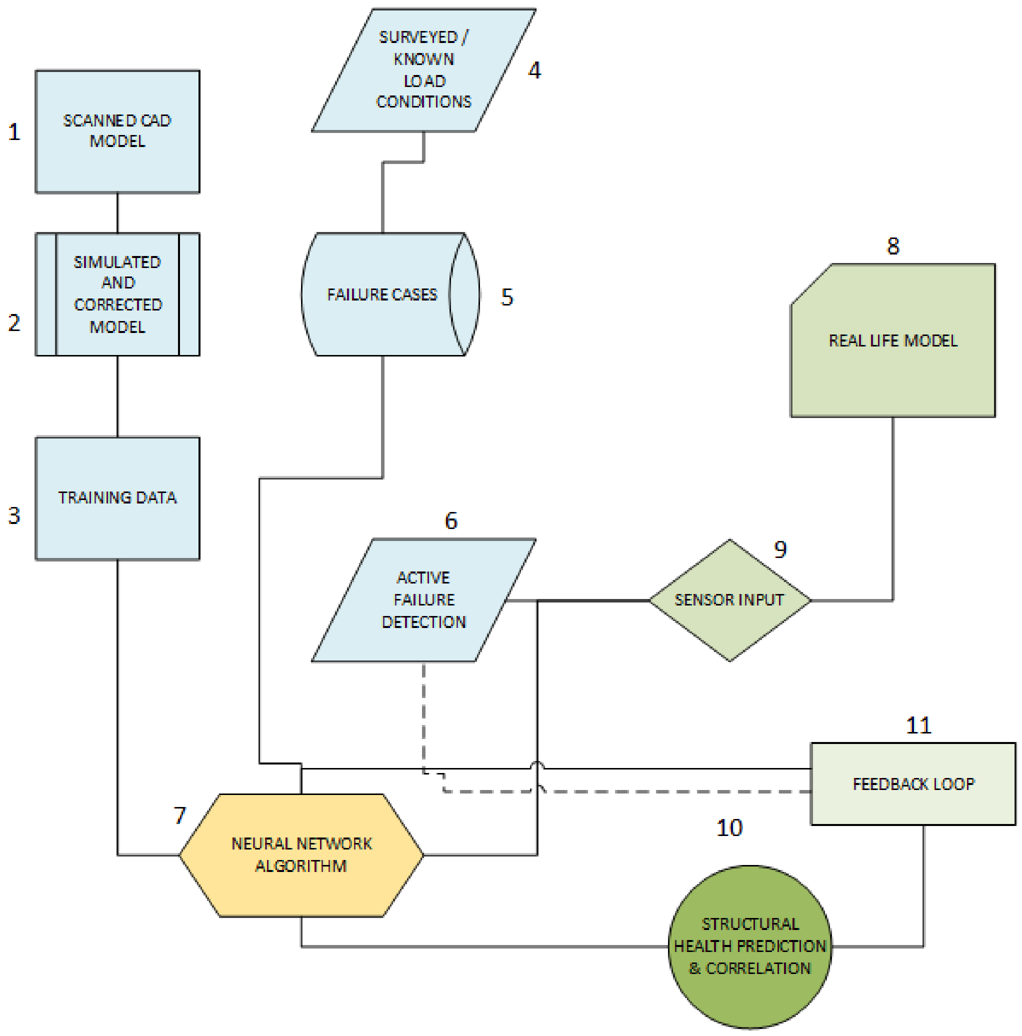


Figure 6 Solution Approach

Proposed Solution

Currently available SHM systems for transportation infrastructure involve cumbersome installations that require structural modifications and cause traffic interruptions. Moreover, the cost of sensors for each structure is relatively high. Consequently, the deployment of

SHM systems is complicated. Understanding and predicting structural health and failure modes are difficult without collecting and analyzing real-time data from the structures. To address this problem, a novel system was designed and tested. The proposed system can be subdivided into the following steps:

Identification and Characterization of the Structure: This is accomplished by 3D scanning of the structure. The structure is then modeled with the help of a computer-aided modeling (CAD) software. These steps also include the collection of data regarding the composition of materials and their mechanical characteristics.

Computational Analysis: This stage involves computer modeling of the specific structure. A computer model is designed, and FE analysis mesh is prepared to simulate the behavior of the structure under different loads and triggered failures.

Load Simulation and Library Generation: This step involves the creation of a library of known failure cases for the structure. Using FE analysis, various loading scenarios are simulated and the results of each are recorded in a library of expected/known structural behaviors.

Sensor Placement: Based on the library of known behaviors and failure modes, a filtering algorithm identifies the structural points with the highest probability of deflection. The algorithm then filters those that are most likely to serve as identifiers of failure scenarios (i.e. points that saw high deflection with high correlation to failure). This yields a list of “high interest” points that will dictate the position of the sensor markers.

Data Collection: Infrared sensors are deployed due to their availability and their ability to be used in a wide range of light conditions. These sensors are collect data on an active structure. The sensor system consists of small infrared-enabled cameras which can track the position of the markers to submillimeter accuracy in conjunction with surface markers placed in “high interest” points. This facilitates the recording of bridge deformation (marker displacement) from a safe distance.

Data Comparison and Prediction: Using a NN, sensor readings are compared in order to know or predict failure scenarios. This step takes advantage of several elements of machine learning algorithms:

- The possibility of using a combination of predicted (calculated) and real data sets for training enables early predictions and increased accuracy as the number of training sets grows.
- The possibility of using simulated failure data sets as part of the training library reduces the need for destructive testing in large structures.
- The ability to interpolate and extrapolate data enables data-based comparisons and predictions, even with a limited number of points.

Chapter 5 Case Study

Case Study Introduction

To test and validate the proposed solution approach, a pilot project was conducted on a model of a steel bridge that resembled a large truss structured bridge in shape and behavior. The model bridge was subjected to all the steps mentioned in the solution approach, including the verifications and validation procedures (such as simulated loading and strain gauges installation for data validation). This specific pilot project yielded promising results, and the system did identify a possible correlation between simulated loading and future failure modes of the structure. Several experiments were conducted in a controlled environment to collect data and validate system inputs.

Considering the large number of truss bridges across the United States [54], a simplified, scaled-down version of a bridge structure was selected for the pilot project. This reduced size allows for a controlled environment suitable for data-collection by using permanently mounted infrared sensors. This arrangement reduces errors due to inaccurate sensor placements or movements. Moreover, the bridge is made of a relatively uniform material, so that intrinsic material properties could be simulated accurately, in this case, steel. The proposed procedure was applied on the model bridge, as explained in the next section.

Development of a 3D Computer Model and Finite Element

The computer model of the bridge was designed with the exact specifications of the model via 3D scans of its geometry. Furthermore, simulations of necessary fasteners and welding joints, as well as material properties, were carefully incorporated. A mesh suitable for the geometry of the object was developed in an ANSYS simulation environment (Figure 7).

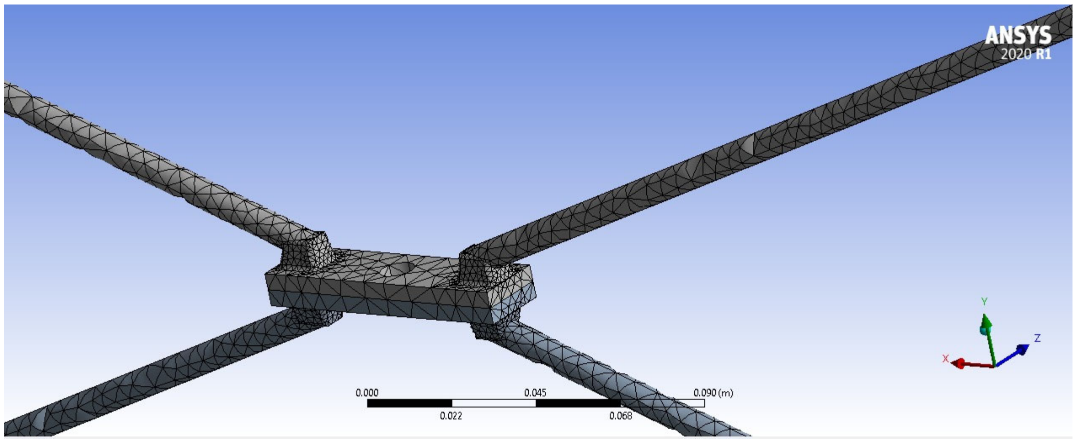


Figure 7 Mesh of the Bridge

The training data set was generated for this pilot project based on the known loading factors for civil engineering infrastructures [55]. Loading conditions were identified and characterized based on the literature review [10], [56], [57]. These conditions were fed into the computer model to generate a data set of expected stresses and strains in the structure of the model.

Load Simulation and Library Generation

By using the computer model, simulations were completed for the deformation of the structure under loading conditions. At any point in which the material passes its elastic deformation region, a material failure can be identified [58]. A gradient boosting neural network used the generated library to detect necessary correlations of input data with known loading conditions and failure modes. Extrapolated input data was also used to match and identify possible relationships to failure scenarios. The machine learning algorithm used all these input data for training in order to enhance accuracy and predictive capacity. The developed model takes into account the material deformation and localized strain as well as the limits of the selected material. Collected data can be subdivided into categories: node (location within the model's coordinate system), material deformation (obtained by the OptiTrack sensors or calculated from strain sensors), and strain (inversely collected by strain sensors or derived from the deformation) [10]. All three types of data were used to train the machine learning algorithm.

In this research, a computational model was developed that can recreate point measurements of essential indicators during different load cases and failure scenarios. The model was adjusted by changing the load and environmental conditions to simulate, trigger, and collect data on the failure modes of loading fatigue, mechanical overload, and creep. A sample case of loading fatigue is shown in Figure 8.

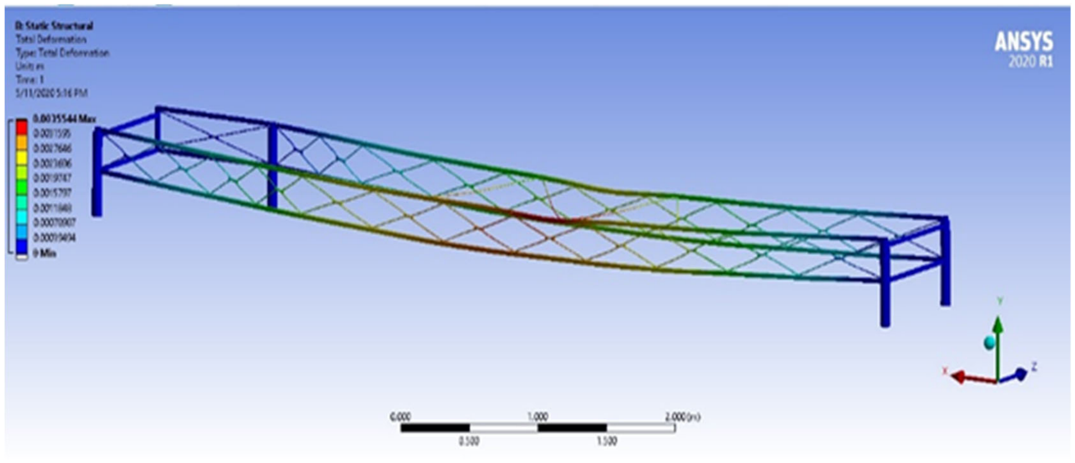


Figure 8 Sample Case for Loading Fatigue

The data for each failure scenario was collected in terms of strain, deformation, and node location. An example of node meshing is shown in Figure 9. The system has over one million nodes or mesh elements; these nodes were considered for the filtering algorithm that determines the best available sensor positions.

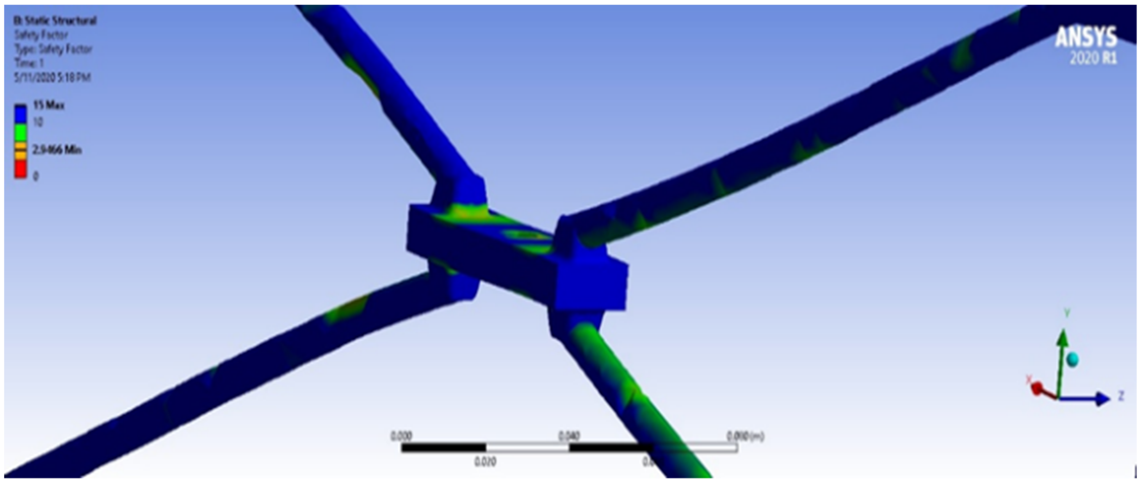


Figure 9 Node Mesh

Smoothing in 2D

Safety Factor vs. Node and Deformation vs. Node were plotted separately (See Figure 10). Both a low pass filter and a moving average filter with sampling were tried. Due to a Gibbs-like phenomenon that occurred with the low pass filter (extrema pushed out of place), the moving average filter with sampling was selected as the most appropriate technique. Hence, data was averaged over every 5,000 nodes, and these averages were plotted to get smoothed curves. The number 5,000 was derived by testing various window sizes and counting the number of resulting extrema. Accordingly, a window size that made the smoothed curve look reasonably noiseless and provided on the order of 100 extrema was selected.

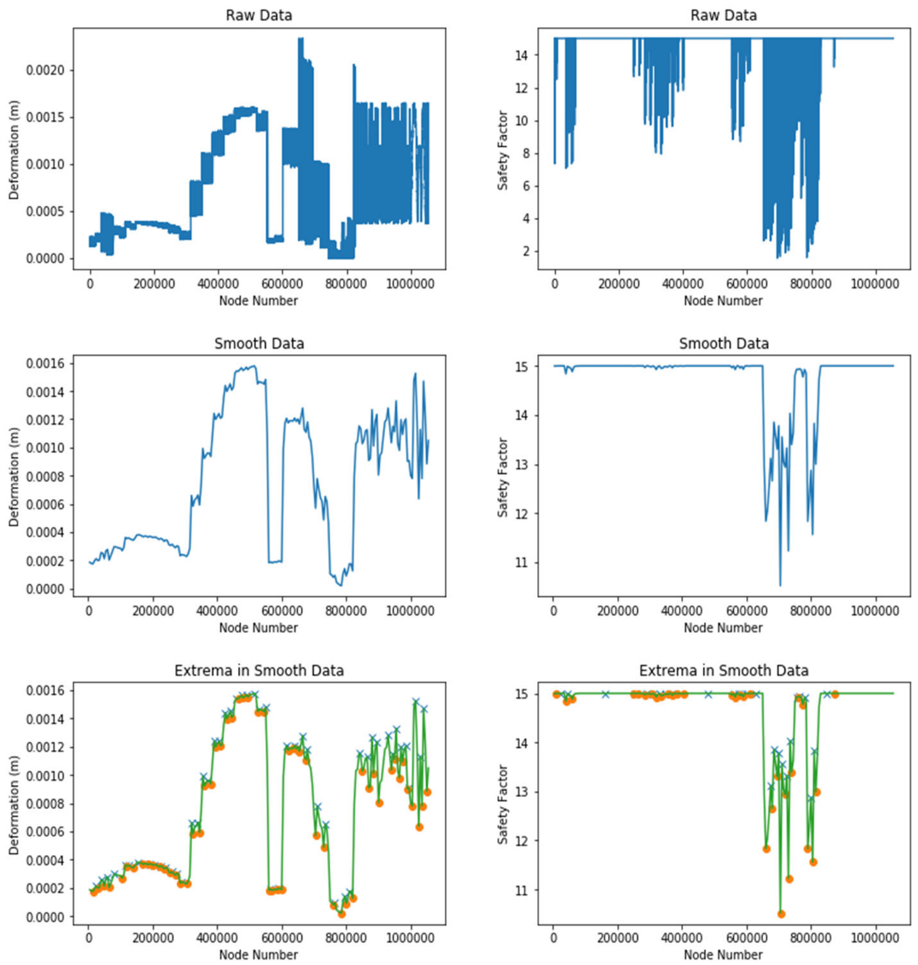


Figure 10 Smoothing in 2D

Smoothing in 3D

Safety factor was plotted on the z-axis and deformation was plotted on the x-axis in two different 3D plots. The corresponding minima and maxima are highlighted (Figure 11). The 3D smoothing is relatively complicated. The smoothed curve is built from B-spline basis functions; in theory, it incorporates the effects of both safety factor and deformation simultaneously while smoothing. However, it did not yield significantly different results. As 3D proved to be a less interpretable method that did not yield better results, 2D methodology was used in this research.

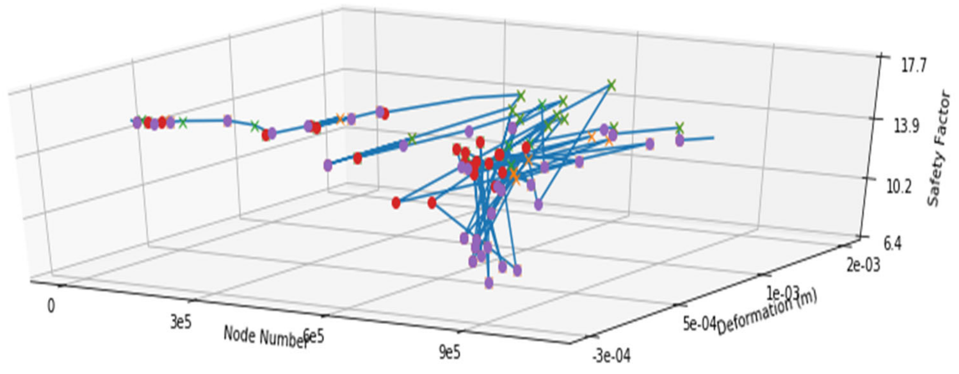


Figure 11 Smoothing in 3D

The data points that are more susceptible to failure were identified. Assuming that material characteristics are uniform, then calculations are performed to determine the points at which the factor of safety indicates a deformation change beyond the expected level for safety infrastructure operation [7]. The algorithm yields the primary location of interest in the model by the node number, deformation, stress, strain, and, ultimately, the safety factor.

The data list contains the inflection points of the simulated structural failure that enable the identification of the position of an optimized sensor array. Based on the number of available sensors, this list is then truncated at an arbitrary number for comparison. In this pilot project, the list of extrema readings was truncated at 100 in order to encompass the most critical and extreme conditions during the load test. These node locations served as the location of the initial set of sensor nodes in both the training data and the experimental setup.

The second array of influential data points is a subset of the original model, with a maximum number of 100 data points reflecting the behavior of 100 coordinates in the structure (simulating these points in terms of the mechanical parameters). It is worth mentioning that the sensor placement algorithm can be modified to accommodate as many sensor arrays as necessary for the desired accuracy.

Optimization of Sensor Locations

In this phase, a safety factor vs. node plot (Figure 12) and a deformation vs. node plot (Figure 13) were generated separately. They represent the complete output of the model under a specified loading condition (e.g., Figure 13 represents a load of 2,500 lbs. in the predetermined position 1). This data set was then used for sampling after filtering by applying a moving average filter. Hence, data over every 5,000 nodes were averaged and then plotted to obtain smoothed curves. The number 5,000 was selected by testing various window sizes and counting the number of extrema. This process resulted in the identification of a window size that made a smoothed curve look reasonably noiseless and gave extrema in the order of 100. It falls within the current channel availability of the selected sensors.

Once the data points that were more indicative of deformation were identified, it was assumed that the material characteristics were uniform. Then, calculations were performed to determine the points at which the factor of safety indicated a deformation change beyond the expected level for safety infrastructure operation. The algorithm yielded the primary location of interest in the model by the node number, deformation, stress, strain, and, ultimately, the safety factor.

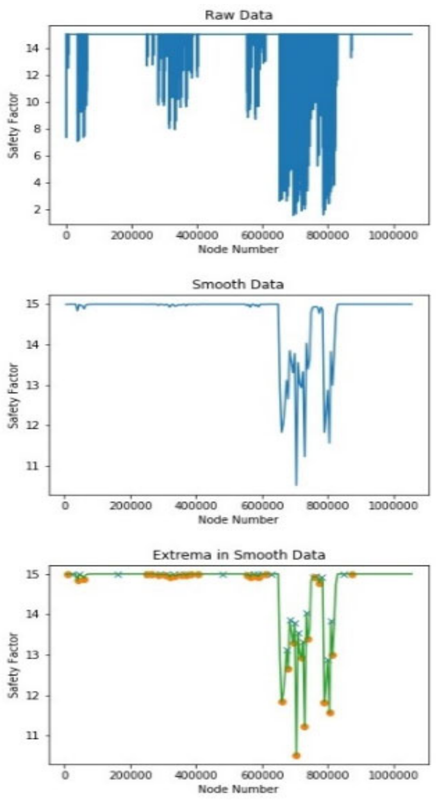


Figure 12 Safety Factor vs. Node

Figure 13 shows that the deformation plot corresponds to the relative maxima for the deformation in 2D. It outperforms a 3D plot in terms of clarity and readability. The smoothed curve was built from B-spline basis functions, and extrema were identified.

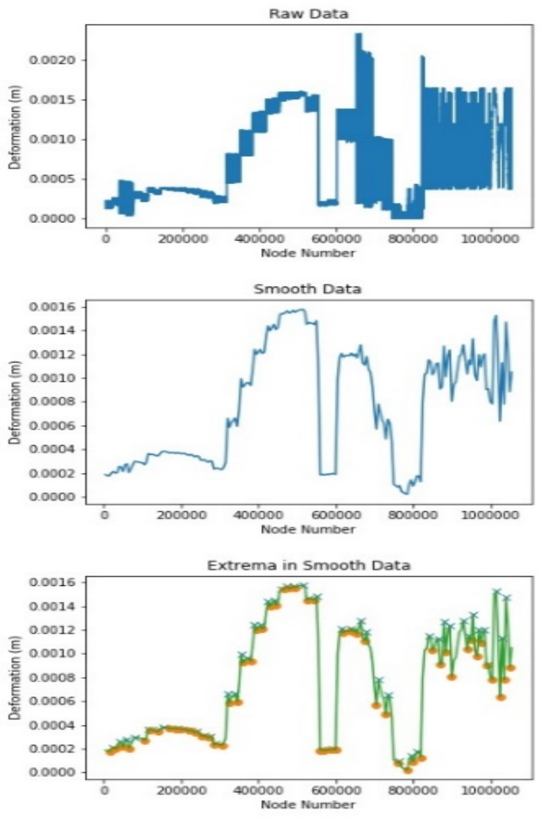


Figure 13 Deformation vs. Node

The data list contains the inflection points of the simulated structural failure that enable the identification of the position of an optimized sensor array. Based on the number of available sensors, the list was then reduced for comparison. In this pilot project, the list of extrema readings was limited to 100, as it encompasses the most critical and extreme conditions during the load test. These node locations were considered in the initial set of sensor nodes in both training and the experimental setup.

The second array of influential data points is a subset of the original model, with a maximum number of 100 data points reflecting the behavior of 100 coordinates in the structure (by simulating these points in terms of the mechanical parameters). It is worth mentioning that the sensor placement algorithm can be modified to accommodate as many sensor arrays as necessary for the desired accuracy.

Process Data for Sensor Placement

Each failure mode yielded a data array containing location value and deformation at the location, creating a library of failure modes and essential mechanical readings. Each array was then processed as a multidimensional matrix. These matrix values correspond to the coordinates and material of a node. Strain was calculated by using the material's internal properties to determine the material deformation in the elastic and plastic regions and any material failure. This matrix was exported and processed by a filtering algorithm to reduce noise, decimate data to avoid duplicated data at nearby locations, and then to find the local extrema.

The model returned a list of local maxima and minima based on the preselected parameters. A threshold for material deformation was added, and only the lowest (minima) points were selected based on a predetermined limit of material safety factor.

Data Collection and Preprocessing

Development of an accurate and versatile computer model with a robust library of failures and loading scenarios is helpful in structural health analysis. However, an experimental setup is required to test and corroborate these results. The implementation of a contactless sensor array can reduce the cost and complexities [59]. The next step was to develop an experimental setup to test the predictive and interpolating capabilities of the machine learning algorithm, as well as the accuracy of the contactless sensor setup in determining the material deformation.

A model of the bridge was placed in an optical track sensor-enabled test room (Figure 14). A set of data markers and strain gauges were attached at the previously identified most critical locations (100), while six strain gauge nodes were installed for systems validation.

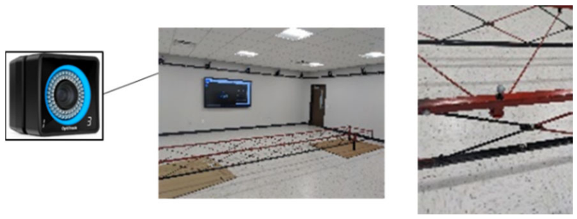


Figure 14 Experimental Setup

The contactless OptiTrack system facilitates the sensor locations within the three-dimensional bridge model to be marked and tracked. Furthermore, it enables collection of real-life deformation data with sub-millimeter accuracy. After the experimental set-up was prepared, a set of experimental data was collected. They were used in gradient boosting neural network to determine the predictive capability. Several mathematical models of the known structures were used for validation of the model, and they demonstrated a trend towards identification of future failure scenarios based on initial data.

The system's data was fed as input data of gradient boosting neural network [14], enabling an interpolation of bridge conditions based on sensor inputs and a correlation with known failure or non-failure cases. The machine learning system was successful in calculating a percentage of correlation to specific sets of known failures. If the input data is identified with a significant correlation with an established failure set, then the known failure location is marked as a point of interest, and the correlation percentage is determined as a predictive indicator of failure.

Defining Failure Modes

Each failure mode yields a data array that contains location value and parameter value (e.g., deformation per location), creating a library of ranges that include each failure mode and essential mechanical readings of each parameter. Each array is then processed as a multidimensional matrix. These matrix values correspond to the coordinates and materials of a node calculated for that location. The strain is calculated and compared with the material's internal properties to determine the material deformation in the elastic and plastic regions, as well as any material failure [43].

This matrix is then exported and processed by a filtering algorithm that reduces noise, decimates the data to avoid duplicated data at nearby locations, and then finds the local extrema points. The model returns a list of local maxima and minima based on the preselected parameters. A threshold for material deformation is added, and only the lowest (minima) points are selected based on a predetermined material safety factor limit. For this research study, any material deformation passed a safety factor of 1 is considered a failure.

Sensor Placement Algorithm

The algorithm aims to collect data from sensors placed at pre-defined positions on the bridges, and failure scenarios are recorded in terms of strain, deformation, and node location. Bridge intrinsic material properties are also considered to determine deformation. The flowchart of the model is presented in figure 15. Every failure mode generates a data array that has the coordinates' value and deformation at the specific points as well as a library of failure modes. This data array is then processed as a multidimensional matrix. The strain is calculated by determining the material's intrinsic properties, as strain is the most important variable, directly affecting the condition of monitoring. Here, our filtering algorithm is in use. It reduces the outliers, removes duplicate data present in the nearby locations, and finds the local extrema in the processed data from sensors.

Considering the pre-selected parameters, the algorithm returns a list of local minima and maxima. The material deformation threshold was already selected based on the material safety factor, so it is added in the model and the lowest minima points are selected. Any deformation in the material higher than the safety factor 1 is considered as a failure in the bridge structure. The ultimate goal of the algorithm is to generate the main points of interest in the model by considering the node number, deformation, stress, strain, and, in the end, the safety factor.

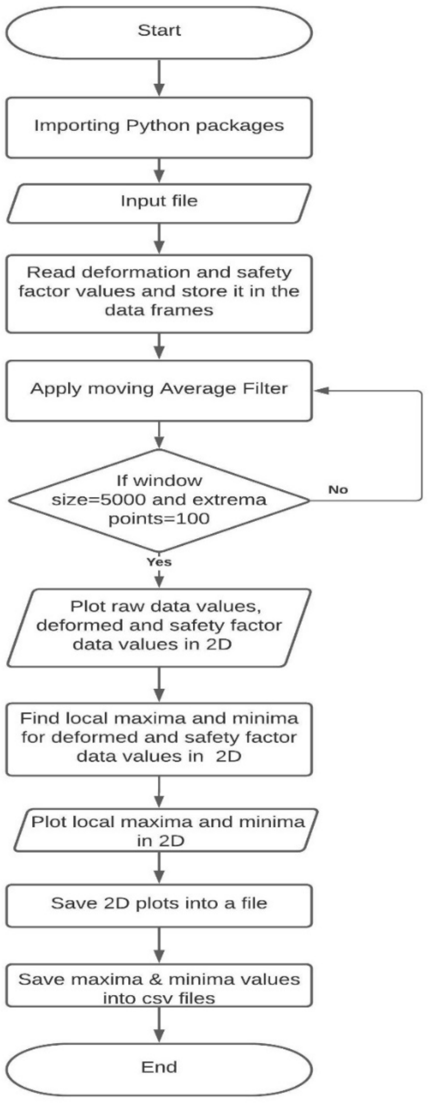

Start
Importing Python packages
Input file
Read deformation and safety factor values and store it in the data frames
Apply moving Average Filter
If window size=5000 and extrema points=100
No
Yes
Plot raw data values, deformed and safety factor data values in 2D
Find local maxima and minima for deformed and safety factor data values in 2D
Plot local maxima and minima in 2D
Save 2D plots into a file
Save maxima & minima values into csv files
End

Figure 15 Sensor Location Algorithm

A moving average filter was applied to the data set after filtering and is then used for sampling. The algorithm plotted smoothed curves by averaging 5,000 nodes. This number was tested over different window sizes and by counting the number of extrema. As such, this became our default window size and produced noiseless, smoothed curves and gave out 100 extrema points.

The inflection points of the simulated structural failure were present in the sensor data array list that identifies the optimized sensor array. During the load test extrema, readings were confined to 100, encompassing the most critical and extreme conditions.

In a 3D model of our approach, the x-axis, y-axis, and z-axis represent node number, safety factor, and deformation, respectively. Firstly, we set all labels for different axis and then found the maxima and minima for deformation and safety factor values. We then plotted the values to visualize the results and saved it in a separate file. Also, we saved the maxima and minima points in an Excel file. Then, we repeated the above process after changing the axis; for instance, deformation values became the y-axis and safety factor values became the z-axis in order to obtain information about failure points in 3D. The smoothed curves were generated from the B-spline basis function, and extrema were identified.

Step 1: Initializing the pre-requisite libraries.

Step 2: Input file containing the data values.

Step 3: Read values and store in data frames.

Step 4: Apply moving average filtering.

Step 5: If window size = 5,000 and extrema points =100. If Yes, go to step 6; if else, return to step 4.

Step 6: Plot raw and smooth values in 2D.

Step 7: Find local maxima and minima of smooth values.

Step 8: Plot local maxima and minima.

Step 9: Save plots.

Step 10: Save maxima and minima data values.

The whole output of the model is represented under the specific loading conditions. Firstly, safety factor vs. node graphs are generated in (Figure 2). Experiments applied on different positions and different weights and their graphs were recorded as well. An initial weight of 2,500 lbs. on position 1 was considered, then the experiment increased the weight incrementally, to 5,000, 7,000, and 10,000 lbs. Graphs were generated, and maxima and minima were determined. Testing was also completed on the same weight, but in different positions, such as 2, 3, and 4. This figure represents a load of 5,000 lbs. in the predetermined position. 4.'x' represent maxima and 'o' represent minima in the data chart. By using a moving average filter, the data set was then sampled after filtering. The data set was averaged over every 5,000 nodes to plot the smoothed curves. This number was selected over different window sizes , producing the correct number of extrema and noiseless smoothed curves.

The value of extrema for this algorithm and window size was selected as 100. Similarly, deformation vs. node graphs are presented in Figure 3. The graphs represent the raw data, smooth data, and extremas and minimas in the smoothed data. As the node numbers increased, the graphs became consistent and uniform. They show the uniform behavior of our algorithm: that after finding enough values by selecting window size 5,000 and extrema points 100, the algorithm identifies the deformation.

To optimize the precise location of the sensors, graphs were generated. Experimental results validated that the average filtering algorithm gives the desired result and is helpful in determining sensor placement in bridges to find the breakage points.

To get a higher level of accuracy, one can increase the sensor arrays, but this can increase the deployment cost. The plots generated are in 2D, but they surpass 3D plots in terms of clarity and readability.

Chapter 6
Validation of Model and Verification

Another experiment was conducted to validate the computer model, and data was collected by using the OptiTrack sensors. A custom set of wireless sensor nodes containing strain gauge sensors was created by adapting a wireless microcontroller to a modified load cell amplifier and attaching it to a set of strain gauges on pre-identified locations within the experimental setup of the structure. By measuring strain at six positions, and by using known loading conditions of 50 kgs, the validity of the computer model and the accuracy of the material deflection detected by the OptiTrack system were determined. Additionally, this data set was used for a cooperative analysis between the data collection of non-contact sensors (OptiTrack system). Figure 15 shows a wiring diagram of the developed sensor circuit. These sensors were placed at six locations.

The sensor data at the first 100 critical points was generated and collected for 1,200 different loading conditions, ranging from wind loading to static loading to seismic simulations [13]. The data obtained from the computer model was used to create a training library for the initialization of the machine-learning algorithm. The algorithm was used for extrapolation and prediction of structural failure in case of unknown initial and/or loading conditions. Furthermore, the training library served as a reference point for known structural behaviors and failure locations. 1,200 numbers of loading scenarios were generated, each representing a different load and position. Loading scenarios were categorized based on the load position and magnitude (e.g., scenario P12500 represents a load of 2,500 lbs. in the predetermined position 1). Figure 16 represents the outcome of a sample scenario P12500; the colors of each mesh element provide a visual guide indicating deformation from the original state.

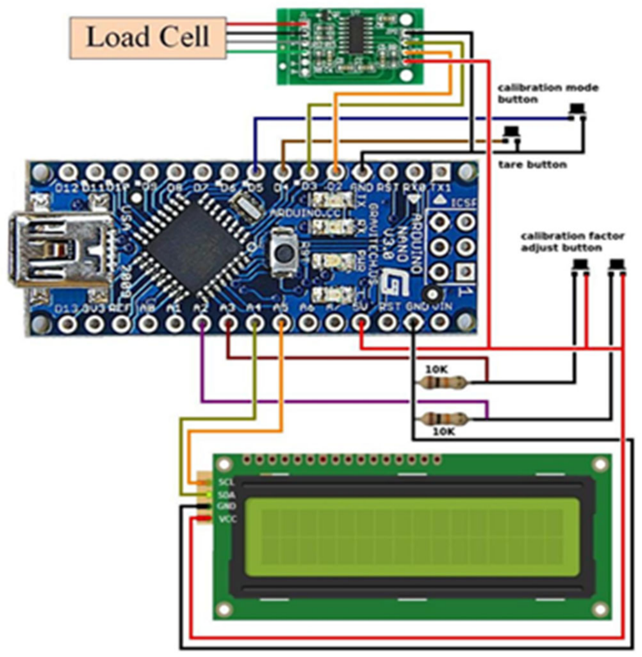


Figure 16 Wiring Diagram for Contact Sensor Nodes

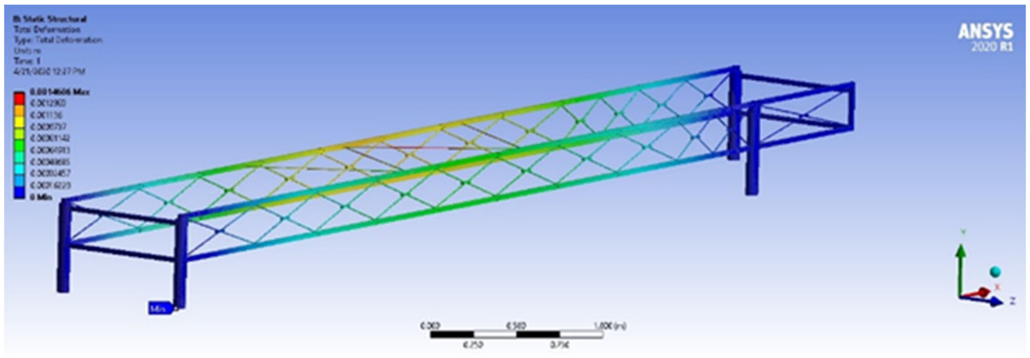


Figure 17 Unloaded Bridge Model

Figure 17 shows the progression of the loading magnitude in order to illustrate the effects of deformation in the structure. When the total strain of mesh elements surpasses the elastic region (or close to the material plastic deformation region), it is marked as a failure point and indicated by a red arrow. This serves as an indicator of structural malfunctions (Figure 18).

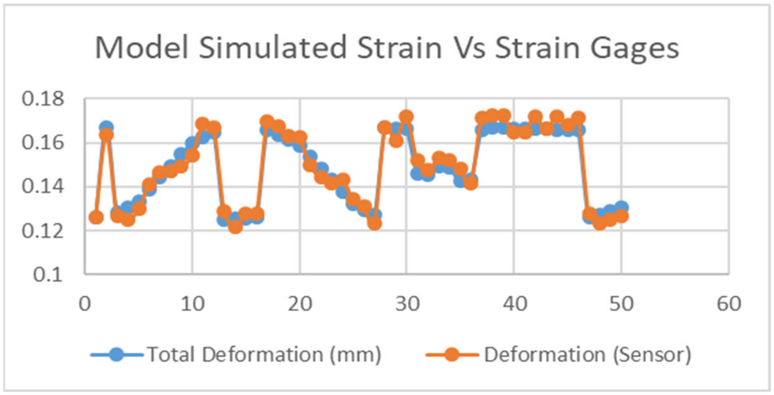


Figure 18 Strain Gauges vs. Computer Model (Simplified)

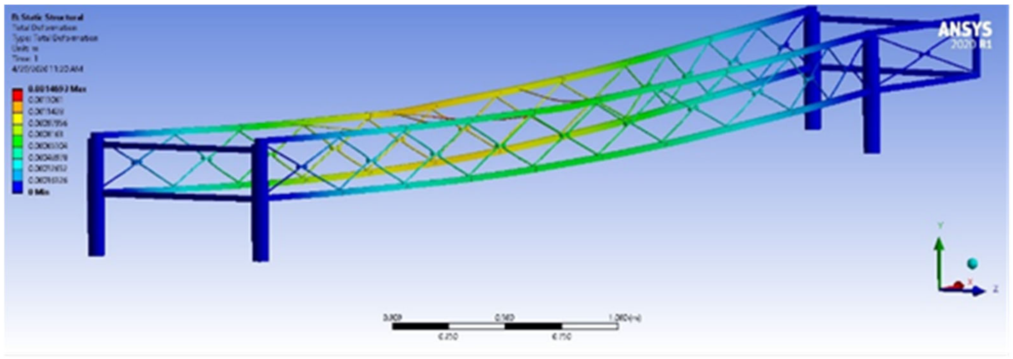


Figure 19 Sample Failure Scenario

Chapter 7 Application of Neural Network Algorithms

In the field of transportation infrastructure, bridges are considered highly important structures. Although the construction of bridges is the costliest among engineering projects related to transportation infrastructure, bridges must be designed in accordance with relevant standard safety measures. However, bridges deteriorate due to aging, mainly in association with the ever-increasing traffic loads [60]. The American Association of State Highway and Transportation Officials (AASHTO) published guidelines for non-destructive evaluation methods, diagnostic load testing, and proof load testing in 2003 [61]. Regardless of testing procedure, the primary objective of any load test is to assess the load capacity more accurately. Due to the drawbacks of simplified analyses based on conservative assumptions, new technologies are being applied in testing and evaluations.

The Diagnostic Load Test, one of the major types of tests used for bridge health monitoring, is performed at controlled load situations and integrates sensors for measurement of response performance [61]. This test focuses on measuring structural deformations for a set of applied load conditions without testing the load capacity directly. The aim of the test is to develop a relationship between the real behavior of the bridge and analytical calculation. It can be used either as an acceptance test of bridge structures or as a tool for assessment of the load carrying capacity of existing bridges [62].

Conducting load tests [63] on different types of bridges, such as concrete bridges [64], reinforced-concrete arch bridges [63], steel bridges [64], masonry arch bridges [65], and stone arch bridges [66], have been documented in the literature. In this research, a model of a steel bridge was constructed in order to perform experiments. Sensors embedded to the bridge, as well as contactless sensors, have been used in research studies related to bridge health monitoring ([8]), [9]), [12]). A contactless OptiTrack system has been used in this experiment for collection of deformation data with sub-millimeter accuracy at the sensor locations. Application of numerous statistical models and machine learning algorithms, including Extreme Gradient Boosting [67], Decision Trees [68], Random Forests [68], and Naive-Bayes for health monitoring of bridge structures, was evident. In order to identify the probability of the deformations, fifteen types of statistical models were created in this research. Coding was conducted in the Python programming language, and the analysis was performed in a Google Collaboratory Notebook. The model development and training were completed using Pycaret, which is a Python-based framework that provides a variety of machine learning tools.

In this research, load-testing experiments were conducted on a model of a bridge (Figure 14). Sensors were positioned on thousands of locations across the model in order to collect deformation data under different loading conditions. Then, the 100 most critical points required to be monitored closely were identified. In this setup, sixteen experiments were conducted, and each experiment was repeated twelve times. As an example, in the first experiment, the bridge was loaded with 2,500 lbs. at position 1, and the deformations were recorded by using 100 sensors. This was then repeated eleven more times. Then, the load was changed to 5,000 lbs., and the process was repeated. Likewise, the experiment was conducted for 04 numbers of load values (2,500 lbs., 5,000 lbs., 7,000 lbs., and 10,000 lbs.). After that, the same process was repeated at three more locations. As a result, 19,200 entries were collected (4 load values x 4 loading positions x 12 times x 100 sensor locations).

The requirement of this experiment was to train a neural network, so that when the bridge was loaded with an unknown weight at an unknown position, the neural network could give the probability of the bridge deformation similar to a known case scenario. Therefore, preparation of collected data for modeling played a vital role.

The labels of the collected data set were numbered from 0 to 15, having sixteen unique labels for the data set collected by conducting sixteen experiments (P15000, P17000, P110000, P22500, P25000, P27000, P210000, P32500, P35000, P37000, P310000, P42500, P45000, P47000, P410000). The notation of the experiment represents the corresponding locations and loads used. For example, the number P47000 relates to the experiment with a load of 7,000 lbs. at position 4.

The decision tree was built to predict the probability of a given feature vector $\{x_i^{Location}, x_i^{Deformation}\}$ of the given set of labels. The decision tree, which is a non-parametric model, was built to predict the failure of the structure. The algorithm optimizes itself approximately compared to an optimal solution in a parametric model (Logistic Regression), as given in equation 1. The decision tree function f_{TREE} is defined as

$$f_{TREE} = \Pr(y = label \mid x) \qquad (1)$$

where Pr is the probability.

The optimization criterion is the log-likelihood interpreted in equation 2.

$$\frac{1}{N}\sum_{i=1}^{N}[y_i \ln f_{TREE}(x_i) + (1 - y_i)\ln(1 - f_{TREE}(x_i))] \qquad (2)$$

The data set with N number of items, which is used to train the neural network, can be represented as

$$\{ (x_i, y_i) \}_{i=1}^{N} \quad (3)$$

where each element x_i of N is called a feature vector, and each element y_i of N is called a label.

A combination (j,t) is found, where j is an input feature and t is a threshold. It selects a feature j and splits the feature into positive and negative sets of examples at threshold t. The combination $(j,\ t)$ that maximizes the entropy H of a split is chosen by applying

$$H(\mathcal{S}_-, \mathcal{S}_+) =^{def} \frac{|\mathcal{S}_-|}{|\mathcal{S}|} H(\mathcal{S}_-) + \frac{|\mathcal{S}_+|}{|\mathcal{S}|} H(\mathcal{S}_+)(4)$$

This experiment is considered to be a supervised learning problem in the field of machine learning. Supervised learning techniques are algorithms that learn from both an input (known as a feature vector) and an output (known as a label). Their objective is to classify a given input into a set of predetermined classes, with the ultimate objective of finding a model that best describes the relationship between the input and the output. This is considered a classification problem in machine learning, as the desired output is a discrete categorical variable instead of a continuous numeric variable, in which the problem would be considered as a regression problem.

In this research, fifteen types of statistical models and machine learning techniques – namely, Cat Boost Classifier, Decision Tree, Extra Trees, K neighbours, Logistic Regression, Ridge Classifier, Linear Discriminant Analysis, Linear Kernel, Naive-Bayes and Quadratic Discriminant Analysis, and five tree-based models – were applied. The tree-based models were Random Forest Classifier, Adaboost, LightGB, Gradient Boosting, and XtremeGB. They handle outliers and features with difference scales particularly well. Extreme Gradient Boosting was the most accurate model for this experiment as per the performance analysis.

Application of Statistical Models and Machine Learning Techniques

Linear Models

Linear models perform well when the problem statement has a relatively linear relationship with the target variable. The Sigmoid function is used to convert predictions into a probability output for classification. It is computationally efficient and highly interpretable and understandable. Two linear models – namely, Logistic Regression and Ridge Classifier (see figure 20) – were applied in this research.

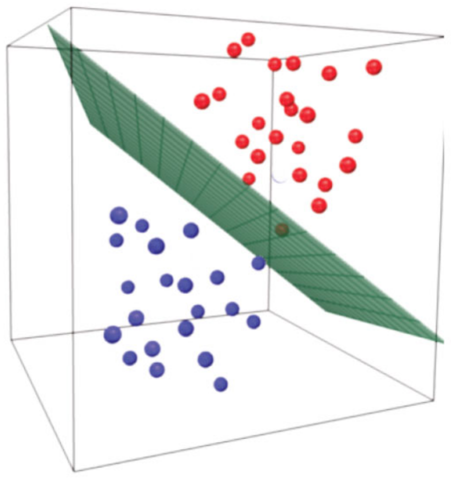


Figure 20 Linear Models

K-Nearest Neighbors (KNN) Models

The KNN model is a non-parametric model (see figure 21) meaning it does not assume a form for the input data compared to models like Logistic Regression. It is, however, a ‘lazy learner', which means it does not learn from the training data used for classification. A KNN Classifier was also applied in this research .

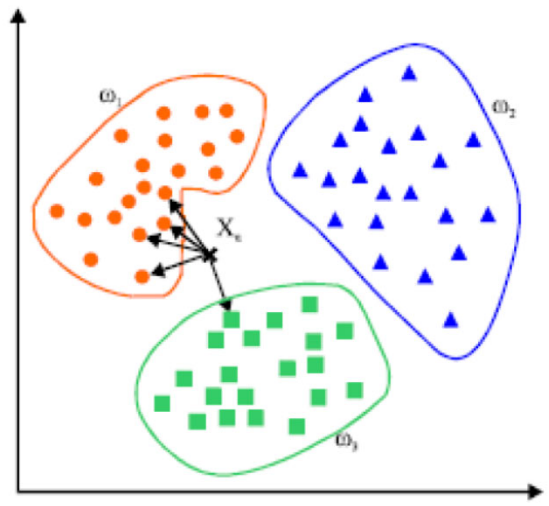


Figure 21 KNN Models

Sequential Neural Network

The training data (augmented images) need to better capture the similarity of each class in a meaningful way. The method used in this exercise is too simplistic and not reliable; the best error rate achieved was 54%, which is very high. Therefore, Sequential Neural Network is not suitable to solve the problem, but it did lay the basis for a possible solution.

Extreme Gradient Boosting Neural Network

An Extreme Gradient Boosting Neural Network model was selected due to its high accuracy score. The model was then developed to correlate new experimental data to computer model simulations of known failure scenarios. The Extreme Gradient Boosting model has been developed based on the Decision Tree algorithm. A decision tree is an acyclic graph that can be used to make decisions. In each branching node of the graph, a specific feature j of the feature vector is examined. If the value of the feature is below a specific threshold, then the left branch is followed; otherwise, the right branch is followed. As the leaf node is reached, the decision is made about the class to which the example belongs [69].

In the Extreme Gradient Boosting model, the sequentially connected ‘learners’ are Decision Trees. Each tree attempts to minimize the error of the previous tree by optimizing on the residuals:

$$[Residuals = y_{predicted} - y_{actual}] \qquad (5)$$

Every time a new tree is added, it fits on a modified version of the initial dataset. These attributes make boosting a highly efficient and accurate model. This process is depicted in Figure 22. The pseudo code of the model pipeline was developed to split the data set into training and testing sets, where 70% of the data was used as the training set, while the remaining 30% was used as the test set. The training set iterate through combinations of a column j and a threshold t that maximizes entropy. The training data were divided into two sets: one set with input values of the column j that is above the threshold t, and the other set below it. The process was repeated until any of the conditions given in the criteria were satisfied, and the test set was evaluated.

Therefore, the algorithm was developed to stop at a leaf node if any of the below mentioned criteria is met.

• All examples in the leaf node are classified correctly by the one-piece model.

• An attribute cannot be identified to split.

• Splitting reduces the entropy less than a specific value (the value is identified experimentally).

• The tree reaches a maximum depth d (the value is identified experimentally).

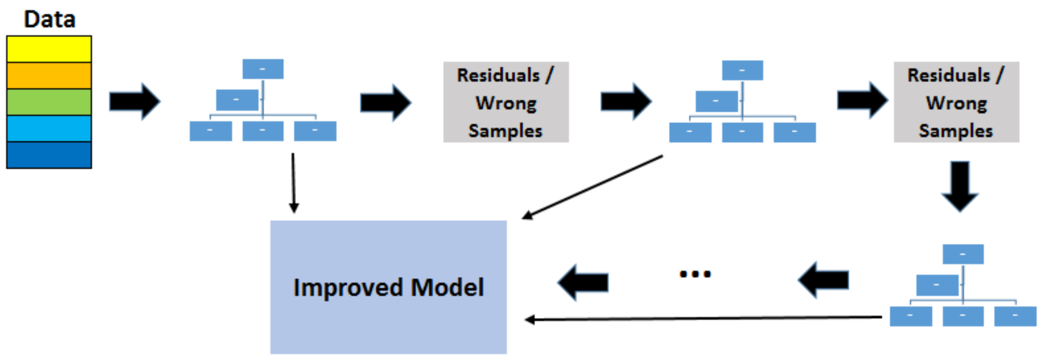


Figure 22 Medium Gradient Boosting Architecture [68]

The first phase in the machine learning pipeline is to process the data in a manner that can be used for model testing. This primarily requires data cleaning, restructuring, feature engineering, and imputation. As this data set was created in a controlled experimental set up, there are no missing values. Furthermore, each class (or label) had an equal number of entries. As a result, there were no imbalances, and each label was equally likely to occur when training a model. Hence, the probability of any example belonging to a certain set remained the same. For example, the probability at the data set P45000 can be mentioned as:

$$Pr(y = P45000) = Pr(y = P47000) = Pr(y = P410000) \quad (6)$$

If there were imbalances, they could cause issues when developing a machine learning model as the model would be trained on data that only presented one label less than 100% of the time. Classifying unseen data that does not belong to that specific class would be difficult as a result.

The data was rearranged into a data frame that could be ingested by a machine learning model, where the input and output variables could be identified in a single tabular format. The top five entries of the data frame are shown in Table 4. The ‘location’ represents the position of the sensor, and the ‘set’ represents the unique location and load applied at each experiment. Each deformation entry has a specific sensor and a set to which it belongs. Every set is present in a single table, which explains why there are multiple entries for a single sensor location.

Table 4: The Top Five Entries of the Data Frame

Index	**location**	**Data set**	**deformation**
0	1	P12500	0.000181
1	1	P12500	0.000189
2	1	P12500	0.000222
3	1	P12500	0.000223
4	1	P12500	0.000223

The average deformation over the 100 locations, when the experiment was repeated twelve times, is illustrated in Figure 23. Each line represents a unique label that corresponds to a load and a position. As four loads were applied at each of the four selected locations, the graph consists of sixteen lines, in which the peaks are very visible. The resulting data sets of the sixteen experiments were ranked based on the average deformation, as depicted in Figure 24. For visualizing the deformation of five such random sets, a three-dimensional plot is presented in Figure 25.

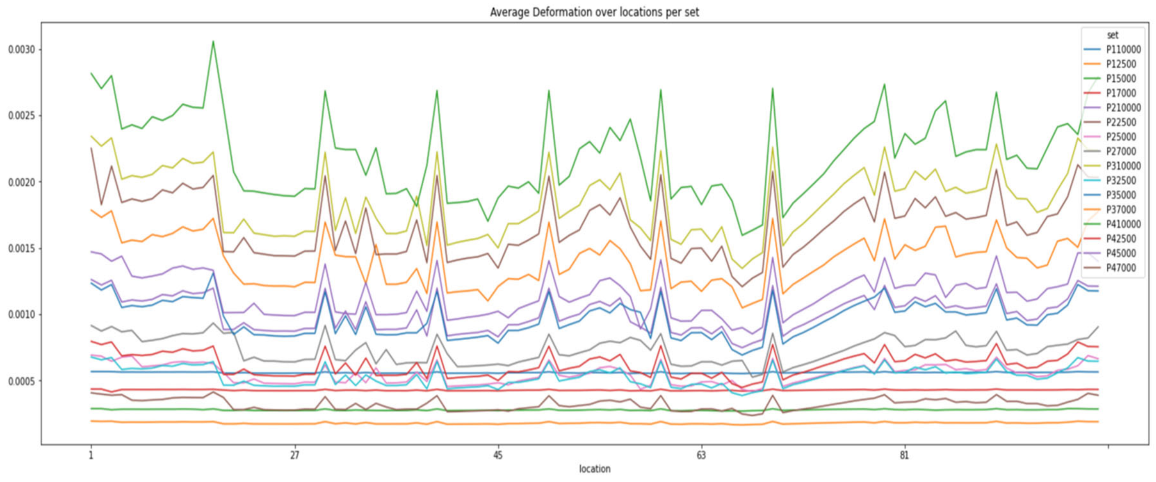


Figure 23 Variation of Average Deformation Over 100 Locations

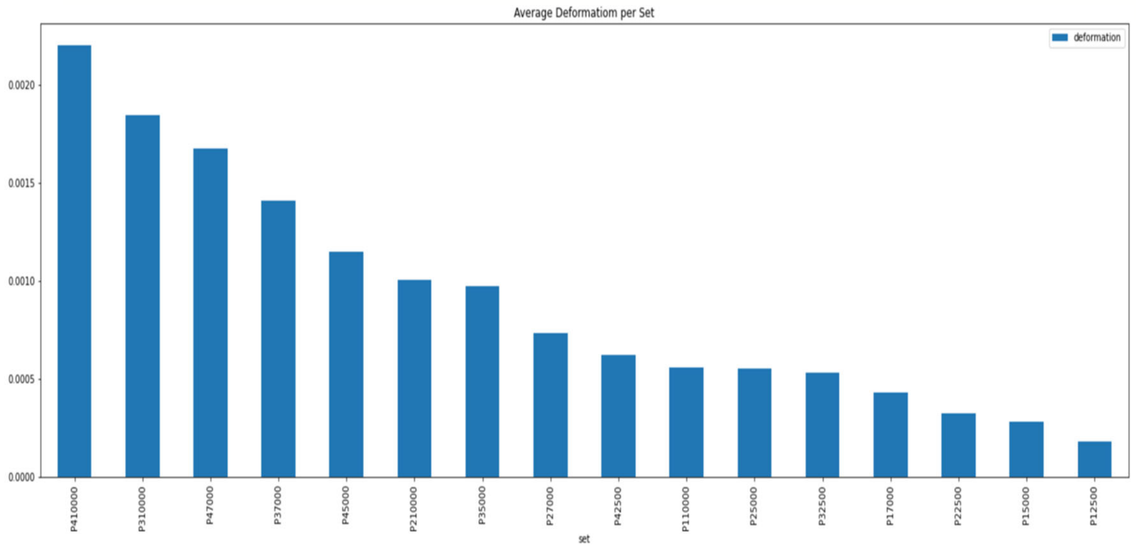


Figure 24 Ranking the Average Deformation Per Set

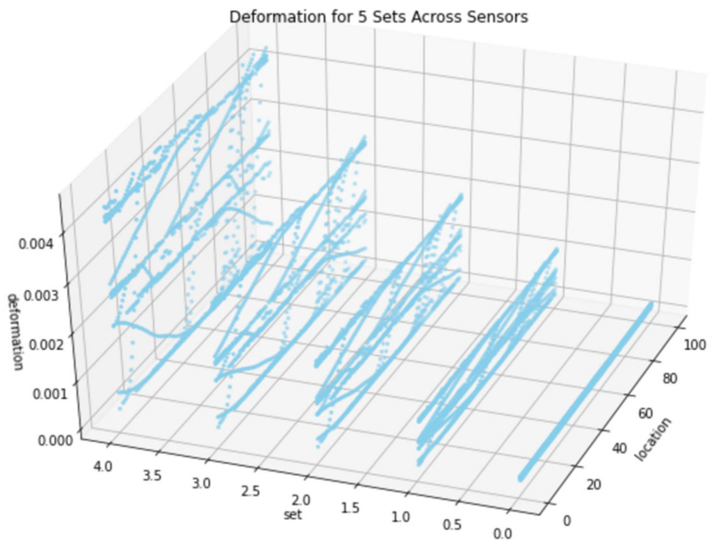


Figure 25 Scatter Plots of Deformation Across Sensors for Five Random Sets

Three data sets (P12500, P27000, P410000) were selected, including the lowest, middle, and highest average deformation values, and the number of occurrences were plotted against deformation. As illustrated in Figure 26, the distribution is clearly a multi-modal pattern for each set.

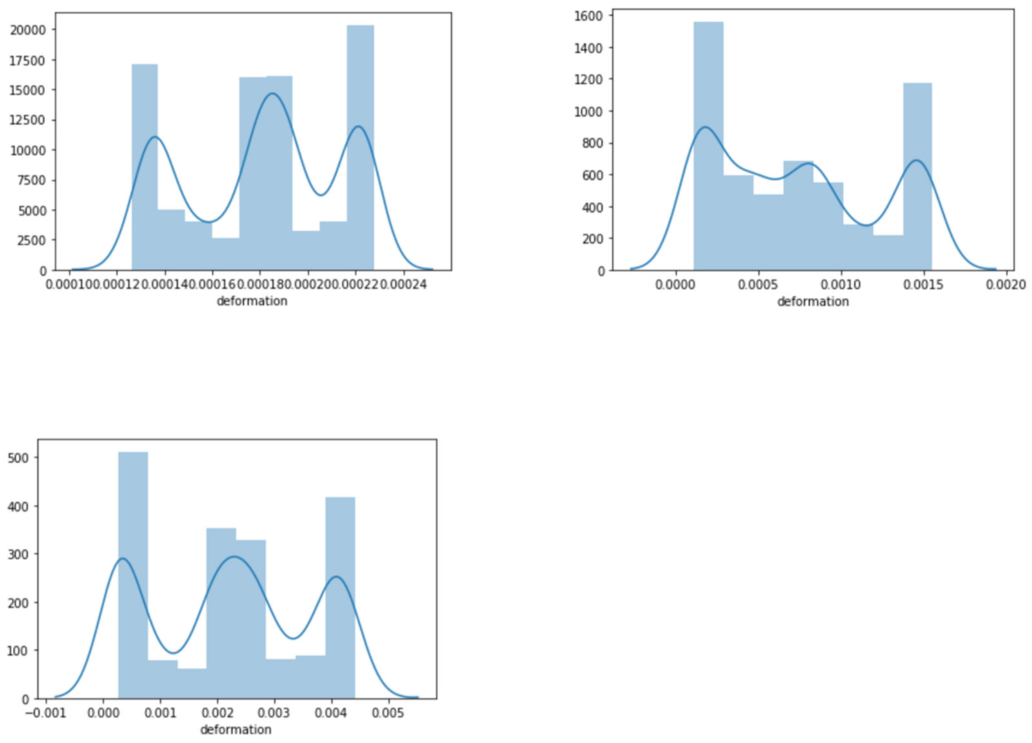


Figure 26 Multi-Modal Distribution Pattern of Data Sets

A code snippet was developed to initialize the setup environment for the Pycaret framework in order to facilitate quick and efficient pipeline creation for machine learning models. The deformation was scaled, for many machine learning algorithms, such as Logistic Regression, Support Vector Machines, K Nearest Neighbours, and Naïve Bayes, assume that all features are centred around zero and have variances that are at the same level of order. This is done using the z-score defined as:

$$\text{z-score:} = \frac{x - \mu}{\sigma} \qquad (7)$$

Data was transformed to modify the shape of the data distribution to have a normal or an approximately normal distribution in order to satisfy certain model assumptions of normality, such as: Logistic Regression, Linear Discriminant Analysis, and Gaussian Naïve Bayes.

This was done using the Box-Cox Transformation

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \lambda \neq 0 \\ \log(y), & if\ \lambda = 0 \end{cases} \qquad (8)$$

where λ varies from -5 to 5, and all values are searched to find the optimal value that best approximates a Normal Distribution. Figure 27 shows the distribution pattern of the data set given in Figure 25 after performing the transformation.

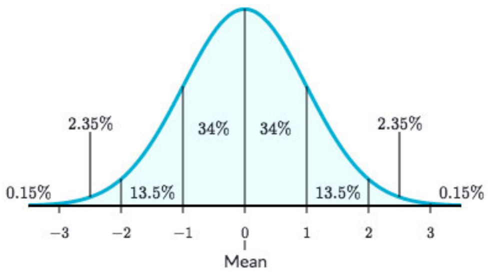


Figure 27 Normal Distribution

Extreme Gradient Boosting is proposed as the best machine learning algorithm as it allows for a stratified analysis of the prediction. Based on the predictions, the engineer can either intuitively associate the input with a certain profile or he or she can create an entirely new profile. In order to calculate and validate the Extreme Gradient Boosting approach, a special library of known data and associated failures was created. In order to identify the efficacy, a set of known loading sets was created in association with one failure set. For example, if a given input deformation set belongs to P47000, 70% of the time we were able to successfully identify that the majority of the set (50% or more) belonged to P47000. When we averaged out five different known sets, we identified that the accuracy of the network to correctly identify the majority of the set belonging is equal to 72%. The appendix in this document includes samples of the testing data sets used for this accuracy calculation.

The developed framework of a predictive SHM system that allows easy installation and assessment of a large structure via non-contact sensors eliminates the need for permanent or complex sensor installation. The system identifies essential critical elements in failure and adjusts them according to the data source of a health status prediction. A mathematical model was established and validated to collect data sets of structural deformation, and then a library was yielded for modeled structure loading scenarios. The model generated over 1,200 load cases, replicating failure and non-failure conditions of the structure of interest based on estimated or known loading cases of similar structures. Gradient boosting neural network was identified as the most suitable machine-learning algorithm, and it was applied in order to extrapolate and correlate data, giving an indication of possible structural damage with an accuracy of 76%.

The proposed system could be used to estimate failure prediction of civil structures, incorporating computer models to train the machine learning algorithm, and to assist in predicting failure in structures that are large and complex, like real-world applications. Additionally, a combination of computer model and machine learning allows prediction of health in a system that has unknown initial conditions, allowing it to omit the loading and fatigue history of bridges and use current deformation data to extrapolate possible failure cases.

Accuracy of the systems may be improved with additional test scenarios, more complete test cases (destructive testing), and with exposure to new loading scenarios. The creation of a set of destructive structural examples to test the accuracy and validity of the prediction in both failure and non-failure scenarios will also be helpful. Deployment of the proposed system in a real small structure will provide a better understanding of how machine learning algorithms are capable of incorporating new loading conditions as the system records them.

Chapter 8 Results and Conclusions

This dissertation presents the framework of a predictive SHM system that allows an easy installation and assessment of a large structure via non-contact sensors, avoiding the need for installation of a sensor system. The system identifies essential critical elements in failures and adjustments according to the source data for prediction of health status. A mathematical model was established and validated by using collected data sets of structural deformation data. Then, a library of comparative cases was yielded for modeled structure loading scenarios. The model was generated by using over 1,200 load cases and replicating failure and non-failure conditions of the structure based on both estimated or known loading cases of similar structures. A suitable machine-learning algorithm was identified and applied in order to extrapolate and correlate data, giving an indication of possible structural damage with an accuracy of 76%. The proposed system could be used to interpret and correlate data while gathering new information, so that it can provide predictive information that includes possible failure modes and locations as well as a percentage of relationships with the known failure cases.

The system facilitates estimating failure prediction of civil structures with the added ability of using computer models to train the machine learning algorithm, and therefore aids in predicting failure in structures that are too large and complex to have real life samples and experiments. Moreover, a combination of computer model and machine learning enables the prediction of health in a system that has an unknown initial condition. It allows the omission of the loading and fatigue history of bridges and the use of current deformation data to extrapolate possible failure cases. Finally, due to the nature of machine learning, the system's accuracy could be improved with additional test scenarios, more complete test cases (destructive testing), and by exposure to new loading scenarios.

There are avenues for further development of the system that require a creation of a set of destructive structural examples to test the accuracy and validity of the prediction in both failure and non-failure scenarios. Furthermore, the deployment of the current system in a smaller structure will provide a better understanding of how the machine learning algorithm is capable of incorporating new loading conditions when the system records them.

The developed models represent the framework for a predictive structural health system that allows for easy installation and assessment of a large structure via non-contact sensors, avoiding the need for installation. The system identifies essential critical elements in failure and adjusts to incorporating this as the data source for a health status prediction. A mathematical model was established and validated to collect data sets of structural deformation data, yielding a library of comparative cases for modeled structures. The model generated over 1,200 load cases, replicating failure and non-failure conditions for the structure based on estimated or known loading cases for similar structures. Finally, the recurrent neural network was able to interpret and correlate the data while learning new information in order to provide predictive information that includes possible failure mode and location as well as a percentage of relationships with known failure cases. System development is still preliminary and will require the creation of a set of destructive structural examples to test the accuracy and validity of the prediction in both failure and non-failure scenarios. Additionally, the deployment of the current system in a full-size smaller structure will provide a better understanding of how the machine learning algorithm is capable of incorporating new loading conditions as the system records them.

References

[1] H. Guo, G. Xiao, N. Mrad, and J. Yao, “Fiber optic sensors for structural health monitoring of air platforms,” *Sensors*, vol. 11, no. 4. pp. 3687–3705, Apr. 2011, doi: 10.3390/s110403687.

[2] J. M. Ko and Y. Q. Ni, “Technology developments in structural health monitoring of large-scale bridges,” *Eng. Struct.*, vol. 27, no. 12 SPEC. ISS., pp. 1715–1725, 2005, doi: 10.1016/j.engstruct.2005.02.021.

[3] H. Guo, G. Xiao, N. Mrad, and J. Yao, “Fiber optic sensors for structural health monitoring of air platforms,” *Sensors*, vol. 11, no. 4, pp. 3687–3705, Apr. 2011, doi: 10.3390/s110403687.

[4] American Society of Civil Engineers, “Report card for America’s infrastructure,” 2017.

[5] F. Moaveni, B., Hurlebaus, S. and Moon, “Special issue on real-world applications of structural identification and health monitoring methodologies,” *J. Struct. Eng*, no. 139(10), pp. 1637–1638, 2013.

[6] T. Adams, T., Mashayekhizadeh, M., Santini-Bell, E., Wosnik, M., Baldwin, K. and Fu, “No TitlStructural Response Monitoring of a Vertical Lift Truss Bridge,” *Transp. Res. Board 96th Annu. Meet.*, vol. (No. 17-06, 2017.

[7] M. J. Rosales and R. Liyanapathirana, “Data driven innovations in structural health monitoring,” in *Journal of Physics: Conference Series*, Jun. 2017, vol. 842, no. 1, doi: 10.1088/1742-6596/842/1/012012.

[8] J. M. López-Higuera, L. R. Cobo, A. Q. Incera, and A. Cobo, “Fiber optic sensors in structural health monitoring,” *J. Light. Technol.*, vol. 29, no. 4, pp. 587–608, 2011, doi: 10.1109/JLT.2011.2106479.

[9] D. H. Kim and M. Q. Feng, “Real-time structural health monitoring using a novel fiber-optic accelerometer system,” *IEEE Sens. J.*, vol. 7, no. 4, pp. 536–543, Apr. 2007, doi: 10.1109/JSEN.2007.891988.

[10] J. D. Doornink, B. M. Phares, T. J. Wipf, and D. L. Wood, “Damage detection in bridges through fiber optic structural health monitoring,” in *Photonic Sensing Technologies*, Oct. 2006, vol. 6371, p. 637102, doi: 10.1117/12.686011.

[11] M. Abdelbarr, Y. L. Chen, M. R. Jahanshahi, S. F. Masri, W. M. Shen, and U. A. Qidwai, “3D dynamic displacement-field measurement for structural health monitoring using inexpensive RGB-D based sensor,” *Smart Mater. Struct.*, vol. 26, no. 12, Nov. 2017, doi: 10.1088/1361-665X/aa9450.

[12] T. and C. Khuc, “Completely contactless structural health monitoring of real-life structures using cameras and computer vision,” *Struct. Control Heal. Monit.*, vol. 24, p. 1852, 2017.

[13] Y. Xu and J. M. W. Brownjohn, “Review of machine-vision based methodologies for displacement measurement in civil structures,” *J. Civ. Struct. Heal. Monit.*, vol. 8, no. 1, pp. 91–110, Jan. 2018, doi: 10.1007/s13349-017-0261-4.

[14] F.-G. Yuan, S. A. Zargar, Q. Chen, and S. Wang, “Machine learning for structural health monitoring: challenges and opportunities,” Apr. 2020, p. 2, doi: 10.1117/12.2561610.

[15] El-Shahat, “Advanced Applications for Artificial Neural Networks.,” *IntechOpen*, 2018.

[16] S. K. and D. I. O. Avci, O. Abdeljaber, “Structural Health Monitoring with Self-Organizing Maps and Artificial Neural Networks,” *Top. Modal Anal. Test.*, pp. 237–246, 2020.

[17] J. P. Lynch, C. R. Farrar, and J. E. Michaels, “Structural health monitoring: Technological advances to practical implementations,” *Proc. IEEE*, vol. 104, no. 8, pp. 1508–1512, 2016, doi: 10.1109/JPROC.2016.2588818.

[18] P. Seventekidis, D. Giagopoulos, A. Arailopoulos, and O. Markogiannaki, “Structural Health Monitoring using deep learning with optimal finite element model generated data,” *Mech. Syst. Signal Process.*, vol. 145, p. 106972, 2020, doi: 10.1016/j.ymssp.2020.106972.

[19] D. Giagopoulos, A. Arailopoulos, V. Dertimanis, C. Papadimitriou, E. Chatzi, and K. Grompanopoulos, “Structural health monitoring and fatigue damage estimation using vibration measurements and finite element model updating,” *Struct. Heal. Monit.*, vol. 18, no. 4, pp. 1189–1206, 2019, doi: 10.1177/1475921718790188.

[20] V. Shahsavari, M. Mashayekhi, M. Mehrkash, and E. Santini-Bell, “Diagnostic testing of a vertical lift truss bridge for model verification and decision-making support,” *Front. Built Environ.*, vol. 5, no. July, pp. 1–19, 2019, doi: 10.3389/fbuil.2019.00092.

[21] and P. M. Ostachowicz, Wieslaw, Rohan Soman, “Optimization of sensor placement for structural health monitoring: A review,” *Struct. Heal. Monit.*, vol. 18, no. 3, pp. 963–988, 2019.

[22] Y. Bao, Z. Tang, H. Li, and Y. Zhang, “Computer vision and deep learning–based data anomaly detection method for structural health monitoring,” *Struct. Heal. Monit.*, vol. 18, no. 2, pp. 401–421, 2019, doi:

10.1177/1475921718757405.

[23] C. Yang, “Sensor placement for structural health monitoring using hybrid optimization algorithm based on sensor distribution index and FE grids.,” *Struct. Control Heal. Monit.*, vol. 25, no. 6, p. 2160, 2018.

[24] and X. G. Yang, Chen, Ke Liang, Xuepan Zhang, “Sensor placement algorithm for structural health monitoring with redundancy elimination model based on sub-clustering strategy,” *Mech. Syst. Signal Process.*, vol. 124, pp. 369–387, 2019.

[25] X. W. Ye, C. Z. Dong, and T. Liu, “A Review of Machine Vision-Based Structural Health Monitoring: Methodologies and Applications,” *J. Sensors*, vol. 2016, 2016, doi: 10.1155/2016/7103039.

[26] P. Xiao, Z. Y. Wu, R. Christenson, and S. Lobo-Aguilar, “Development of video analytics with template matching methods for using camera as sensor and application to highway bridge structural health monitoring,” *J. Civ. Struct. Heal. Monit.*, vol. 10, no. 3, pp. 405–424, 2020, doi: 10.1007/s13349-020-00392-6.

[27] G. F. Gomes, F. A. de Almeida, P. da Silva Lopes Alexandrino, S. S. da Cunha, B. S. de Sousa, and A. C. Ancelotti, “A multiobjective sensor placement optimization for SHM systems considering Fisher information matrix and mode shape interpolation,” *Eng. Comput.*, vol. 35, no. 2, pp. 519–535, 2019, doi: 10.1007/s00366-018-0613-7.

[28] E. B. Flynn and M. D. Todd, “A Bayesian approach to optimal sensor placement for structural health monitoring with application to active sensing,” *Mech. Syst. Signal Process.*, vol. 24, no. 4, pp. 891–903, 2010, doi: 10.1016/j.ymssp.2009.09.003.

[29] S. Surya and R. Ravi, “Deployment of Backup Sensors in Wireless Sensor Networks for Structural Health Monitoring,” *Proc. 2nd Int. Conf. Trends Electron. Informatics, ICOEI 2018*, no. Icoei, pp. 1526–1533, 2018, doi: 10.1109/ICOEI.2018.8553680.

[30] D. Dinh-Cong, H. Dang-Trung, and T. Nguyen-Thoi, “An efficient approach for optimal sensor placement and damage identification in laminated composite structures,” *Adv. Eng. Softw.*, vol. 119, no. February, pp. 48–59, 2018, doi: 10.1016/j.advengsoft.2018.02.005.

[31] R. J. Barthorpe and K. Worden, “Emerging Trends in Optimal Structural Health Monitoring System Design: From Sensor Placement to System Evaluation,” *J. Sens. Actuator Networks*, vol. 9, no. 3, p. 31, 2020, doi: 10.3390/jsan9030031.

[32] A. Downey, C. Hu, and S. Laflamme, “Optimal sensor placement within a hybrid dense sensor network using an adaptive genetic algorithm with learning gene pool,” *Struct. Heal. Monit.*, vol. 17, no. 3, pp. 450–460, 2018, doi: 10.1177/1475921717702537.

[33] Z. Chen, X. Zhou, X. Wang, L. Dong, and Y. Qian, “Deployment of a smart structural health monitoring system for long-span arch bridges: A review and a case study,” *Sensors (Switzerland)*, vol. 17, no. 9, 2017, doi: 10.3390/s17092151.

[34] L. Li, G. Liu, L. Zhang, and Q. Li, “Sensor fault detection with generalized likelihood ratio and correlation coefficient for bridge SHM,” *J. Sound Vib.*, vol. 442, pp. 445–458, 2019, doi: 10.1016/j.jsv.2018.10.062.

[35] P. Pachón, R. Castro, E. García-Macías, V. Compan, and E. Puertas, "E. Torroja's bridge: Tailored experimental setup for SHM of a historical bridge with a reduced number of sensors," *Eng. Struct.*, vol. 162, no. January, pp. 11–21, 2018, doi: 10.1016/j.engstruct.2018.02.035.

[36] Z. Ismail, S. Mustapha, M. A. Fakih, and H. Tarhini, "Sensor placement optimization on complex and large metallic and composite structures," *Struct. Heal. Monit.*, vol. 19, no. 1, pp. 262–280, 2020, doi: 10.1177/1475921719841307.

[37] Y. Tan and L. Zhang, "Computational methodologies for optimal sensor placement in structural health monitoring: A review," *Struct. Heal. Monit.*, vol. 19, no. 4, pp. 1287–1308, 2020, doi: 10.1177/1475921719877579.

[38] G. F. Gomes, S. S. da Cunha, P. da Silva Lopes Alexandrino, B. Silva de Sousa, and A. C. Ancelotti, "Sensor placement optimization applied to laminated composite plates under vibration," *Struct. Multidiscip. Optim.*, vol. 58, no. 5, pp. 2099–2118, 2018, doi: 10.1007/s00158-018-2024-1.

[39] M. Valinejadshoubi, A. Bagchi, and O. Moselhi, "Managing Structural Health Monitoring Data Using Building Information Modelling," pp. 22–24, 2016.

[40] M. Flah, I. Nunez, W. Ben Chaabene, and M. L. Nehdi, “Machine Learning Algorithms in Civil Structural Health Monitoring: A Systematic Review,” *Arch. Comput. Methods Eng.*, no. 0123456789, 2020, doi: 10.1007/s11831-020-09471-9.

[41] C. M. Chang, T. K. Lin, and C. W. Chang, “Applications of neural network models for structural health monitoring based on derived modal properties,” *Meas. J. Int. Meas. Confed.*, vol. 129, no. March, pp. 457–470, 2018, doi: 10.1016/j.measurement.2018.07.051.

[42] and J. W. Smarsly, Kay, Kosmas Dragos, “Machine learning techniques for structural health monitoring,” *Eur. Work. Struct. Heal. Monit.*, no. EWSHM 2016), pp. 5–8, 2016.

[43] J. Vitola, D. Tibaduiza, M. Anaya, and F. Pozo, “Structural Damage detection and classification based on Machine learning algorithms,” *8th Eur. Work. Struct. Heal. Monit. EWSHM 2016*, vol. 4, no. July, pp. 2853–2862, 2016.

[44] A. Ibrahim, A. Eltawil, Y. Na, and S. El-Tawil, “A Machine Learning Approach for Structural Health Monitoring Using Noisy Data Sets,” *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 2, pp. 900–908, 2020, doi: 10.1109/TASE.2019.2950958.

[45] J. Peng, S. Zhang, D. Peng, and K. Liang, "Application of machine learning method in bridge health monitoring," *2017 2nd Int. Conf. Reliab. Syst. Eng. ICRSE 2017*, no. Icrse, 2017, doi: 10.1109/ICRSE.2017.8030793.

[46] G. Chen, H. He, Y. Liu, M. Gao, J. Li, and H. Ren, "A Bridge Health Diagnosis Approach Base on Deep Neural Networks," *Proc. - 2019 2nd Int. Conf. Saf. Prod. Informatiz. IICSPI 2019*, pp. 218–222, 2019, doi: 10.1109/IICSPI48186.2019.9095882.

[47] G. Fan, J. Li, and H. Hao, "Dynamic response reconstruction for structural health monitoring using densely connected convolutional networks," *Struct. Heal. Monit.*, 2020, doi: 10.1177/1475921720916881.

[48] D. C. Feng, Z. T. Liu, X. D. Wang, Z. M. Jiang, and S. X. Liang, "Failure mode classification and bearing capacity prediction for reinforced concrete columns based on ensemble machine learning algorithm," *Adv. Eng. Informatics*, vol. 45, no. February 2019, p. 101126, 2020, doi: 10.1016/j.aei.2020.101126.

[49] V. Kailkhura, S. Aravindh, S. S. Jha, and N. Jayanthi, "Ensemble learning-based approach for crack detection using CNN," no. Icoei, pp. 808–815, 2020, doi: 10.1109/icoei48184.2020.9143035.

[50] D. K. Thai, T. M. Tu, T. Q. Bui, and T. T. Bui, “Gradient tree boosting machine learning on predicting the failure modes of the RC panels under impact loads,” *Eng. Comput.*, no. 0123456789, 2019, doi: 10.1007/s00366-019-00842-w.

[51] H. Zargar, K. L. Ryan, and J. D. Marshall, “Feasibility study of a gap damper to control seismic isolator displacements in extreme earthquakes,” *Struct. Control Heal. Monit.*, vol. 20, no. 8, pp. 1159–1175, Aug. 2013, doi: 10.1002/stc.1525.

[52] T. H. T. Chan *et al.*, “Fiber Bragg grating sensors for structural health monitoring of Tsing Ma bridge: Background and experimental observation,” *Eng. Struct.*, vol. 28, no. 5, pp. 648–659, Apr. 2006, doi: 10.1016/j.engstruct.2005.09.018.

[53] R. S. Pressman, *Software Engineering: A Practitioner’s Approach*. 1987.

[54] J. M. W. Brownjohn, “Structural health monitoring of civil infrastructure,” *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 365, no. 1851, pp. 589–622, Feb. 2007, doi: 10.1098/rsta.2006.1925.

[55] D. Zuo and C. W. Letchford, “Wind-induced vibration of a traffic-signal-support structure with cantilevered tapered circular mast arm,” *Eng. Struct.*, vol. 32, no. 10, pp. 3171–3179, Oct. 2010, doi: 10.1016/j.engstruct.2010.06.005.

[56] A. T. Papagiannakis and N. C. Jackson, “Traffic Data Collection Requirements for Reliability in Pavement Design,” doi: 10.1061/ASCE0733-947X2006132:3237.

[57] R. A. Cook, D. Bloomquist, D. S. Richard, and M. A. Kalajian, “DAMPING OF CANTILEVERED TRAFFIC SIGNAL STRUCTURES.”

[58] K. Worden and G. Manson, “The application of machine learning to structural health monitoring,” *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 365, no. 1851, pp. 515–537, Feb. 2007, doi: 10.1098/rsta.2006.1938.

[59] M. Petrovic *et al.*, “Intensity Fiber-Optic Sensor for Structural Health Monitoring Calibrated by Impact Tester,” *IEEE Sens. J.*, vol. 16, no. 9, pp. 3047–3053, May 2016, doi: 10.1109/JSEN.2016.2524045.

[60] M. H. Faber, D. V. Val, and M. G. Stewart, “Proof load testing for bridge assessment and upgrading,” *Eng. Struct.*, vol. 22, no. 12, pp. 1677–1689, 2000, doi: 10.1016/S0141-0296(99)00111-X.

[61] B. Commander, “Evolution of bridge diagnostic load testing in the USA,” *Front. Built Environ.*, vol. 5, p. 57, 2019.

[62] J. R. Olaszek, P., Łagoda, M., & Casas, “Diagnostic load testing and assessment of existing bridges: examples of application,” *Struct. Infrastruct. Eng.*, vol. 10, no. 6, pp. 834–842, 2014.

[63] M. D. Armendariz, R. R., & Bowman, “Improved load rating of an open-spandrel reinforced-concrete arch bridge,” *J. Perform. Constr. Facil.*, vol. 32, no. 4, 2018.

[64] S. Albraheemi, M. J. A., Davids, W. G., Schanck, A., & Tomlinson, “Evaluation and rating of older non-composite steel girder bridges using field live load testing and nonlinear finite element analysis,” *Bridg. Struct.*, vol. 15, pp. 27–41, 2019.

[65] A. Ataei, S., Tajalli, M., & Miri, “Assessment of load carrying capacity and fatigue life expectancy of a monumental masonry arch bridge by field load testing: A case study of Veresk.,” *Struct. Eng. Mech.*, vol. 59, no. 4, pp. 703–718, 2016.

[66] B. Conde, B., Matos, J. C., Oliveira, D. V., & Riveiro, “Probabilistic-based structural assessment of a historic stone arch bridge,” *Struct. Infrastruct. Eng.*, pp. 1–13, 2020.

[67] A. Burkov, *The hundred-page machine learning book (Vol. 1)*. 2019.

[68] S. Yildirim, *Random Forests vs Gradient Boosted Decision Trees*. 2020.

[69] V. K. Vemuri, *The Hundred-Page Machine Learning Book:* 2020.

Appendix

Preprocessing Data

```
[1]: ### Import necessary python packages

     ## Data science packages to more easily work with data
     import pandas as pd
     import numpy as np

     # Plotting packages to visualize data
     import matplotlib.pyplot as plt
     import seaborn as sns
     from mpl_toolkits import mplot3d
     import matplotlib.ticker as mtick

     ## Packages needed to smooth data and find minima/maxima
     from scipy import signal
     from scipy.signal import argrelextrema
     from scipy.signal import find peaks
     from scipy.interpolate import splprep
     import math

     ## Plotting packages in 3d
     from mpl_toolkits.mplot3d import Axes3D
```

Smoothing in 2D

```
[2]: filename = 'P1_2500'
```

```
[3]: ## Read in deformation data from directory and store into a dataframe called df

       df = pd.read_csv(str(filename) + '_deformation.txt',
                                 delimiter = '\t', names =␣
      ↪['node', 'deform'], header =
      0).set_index('node')
```

```
[4]: ### Read in safety factor data from directory and store into a dataframe called␣
      ↪sf

     sf = pd.read_csv(str(filename) + '_safetyfactor.txt',
     delimiter = '\t', names␣
      ↪=['node', 'safety'], header =
      0).set_index('node')
```

```
### SAFETY FACTOR
## Plot original safety factor data (no smoothing)
ax[0,1].plot(sf) ax[0,1].set(xlabel = 'Node
Number', ylabel = 'Safety Factor', title = 'Raw␣
↪Data');

## Plot smoothed deformation data using moving
averages ax[1,1].plot(sf_rolling.node,
sf_rolling.safety) ax[1,1].set(xlabel = 'Node
Number', ylabel = 'Safety Factor', title = 'Smooth␣
↪Data');

## Find local maxima/minima in
safety factor data peaks_sf, _ =
find_peaks(sf_rolling.safety)
valleys_sf, _ =
find_peaks(sf_rolling.safety * -
1)

## Plot local maxima/minima
in safety factor data ##
Maxima plotted with an "x",
minima with an "o"
ax[2,1].plot(sf_rolling.node[peaks_sf],
sf_rolling.safety[peaks_sf], "x")
ax[2,1].plot(sf_rolling.node[valleys_sf],
sf_rolling.safety[valleys_sf], "o")
ax[2,1].plot(sf_rolling.node, sf_rolling.safety)
ax[2,1].set(xlabel = 'Node Number', ylabel = 'Safety
Factor', title = 'Extrema␣ ↪in Smooth Data');


## Save figure to file
fig.savefig('deliv/' + 'plots_2D/' + str(filename) +
'_2D.jpg')
```

```
[7]: # Save local maxima and minima of smoothed data
     finder = {}

     finder['peaks_df_ma'] = list(zip(df_rolling.node[peaks_df], df_rolling.
      ↪deform[peaks_df]))
     finder['valleys_df_ma'] = list(zip(df_rolling.node[valleys_df], df_rolling.
      ↪deform[valleys_df]))

     finder['peaks_sf_ma'] =
     list(zip(sf_rolling.node[peaks_sf], sf_rolling.
      ↪safety[peaks_sf])) finder['valleys_sf_ma'] =
     list(zip(sf_rolling.node[valleys_sf], sf_rolling.
     ↪safety[valleys_sf]))
```

```
[8]: ### Output saved values (maxima and minima) to csv for
2d maxima/minima

     ## Create output filename for maximum of deformations
     in 2d
     f= open('deliv/' + 'extrema_2D/' + str(filename) +
     "_2D_deform_max.csv", 'w+')

     ## Write to file
     f.write("Node
     Number\tDeformation
     (m)\n") output =
     finder['peaks_df_ma']
     for i in
     range(len(output)):
            f.write(str(output[i][0]) + '\t' +
            str(output[i][1]) + '\n')


     ## Create output filename for minimum of deformations
     in 2d
     f= open('deliv/' + 'extrema_2D/' + str(filename) +
     "_2D_deform_min.csv", 'w+')

     ## Write to file
     f.write("Node
     Number\tDeformation
(m)\n") output =
finder['valleys_df_ma
'] for i in
range(len(output)):
       f.write(str(output[i][0]) + '\t' +
       str(output[i][1]) + '\n')


## Create output filename for maximum of safety factor
in 2d
f= open('deliv/' + 'extrema_2D/' + str(filename) +
"_2D_safety_max.csv", 'w+')

## Write to file
f.write("Node
Number\tSafety
Factor\n") output =
finder['peaks_sf_ma
'] for i in
range(len(output)):
       f.write(str(output[i][0]) + '\t' +
       str(output[i][1]) + '\n')

## Create output filename for minimum of safety factor
in 2d
f = open('deliv/' + 'extrema_2D/' + str(filename) +
"_2D_safety_min.csv", 'w+')

## Write to file
f.write("Node
Number\tSafety
Factor\n") output =
finder['valleys_sf_
ma'] for i in
range(len(output)):
   f.write(str(output[i][0]) + '\t' +

str(output[i][1]) + '\n') f.close()
```

Smoothing in 3D

```
[9]: ## Firstly, with safety factor on the y-axis and
deformation on the z-axis.

     cb = pd.merge(df, sf, left_index = True,
     right_index = True) results =
     splprep([cb.index, cb.safety, cb.deform], s
     = 1000000000)

[10]: ## Create current
     figure fig =
     plt.figure(figsize=plt.f
     igaspect(0.25)) ax =
     fig.gca(projection='3d')
     ax.plot(results[0][1][0], results[0][1][1],
     results[0][1][2])

     ## Set x (node number)
     labels ax.set_xlabel('Node
     Number', rotation = 150);
     ax.set_xticks([0, 300000,
     600000, 900000])
     ax.set_xticklabels([0,
     '3e5','6e5','9e5'])


     ## Set y (safety factor)
     labels y_max =
     np.max(results[0][1][1])
     y_min =
     np.min(results[0][1][1])
     ax.set_ylabel('Safety
     Factor', rotation = 100);
     ax.set_yticks(np.linspace(
     y_min, y_max, 4))
     ax.yaxis.set_major_formatter(mtick.FormatStrFormat
     ter('%.1f'))

     ## Set z (deformation)
     labels z_max =
     np.max(results[0][1][2
     ]) z_min =
     np.min(results[0][1][2
     ])
     ax.set_zticks(np.linsp
     ace(z_min, z_max, 4))
     ax.zaxis.set_major_formatter(mtick.FormatStrFormat
     ter('%.e'))
```

[11]:
```
### Save maximum and minimum values:
finder['peaks_sf_y'] =
list(zip(results[0][1][0][peaks_y],␣
↪results[0][1][1][peaks_y],
results[0][1][2][peaks_y])) finder['peaks_df_z'] =
list(zip(results[0][1][0][peaks_z],␣
↪results[0][1][1][peaks_z],
results[0][1][2][peaks_z])) finder['valleys_sf_y']
= list(zip(results[0][1][0][valleys_y],␣
↪results[0][1][1][valleys_y],
results[0][1][2][valleys_y]))
finder['valleys_df_z'] =
list(zip(results[0][1][0][valleys_z],␣
↪results[0][1][1][valleys_z],
 results[0][1][2][valleys_z]))
```

[12]:
```
## Next, with deformation on the y-axis and safety
factor on the z-axis.

results = splprep([cb.index, cb.deform,
cb.safety], s = 1000000000)
```

[13]:
```
## Create current
figure fig =
plt.figure(figsize=plt.f
igaspect(0.25)) ax =
fig.gca(projection='3d')
ax.plot(results[0][1][0], results[0][1][1],
results[0][1][2])
```

```
## Set x (node number)
labels ax.set_xlabel('Node
Number', rotation = 150);
ax.set_xticks([0, 300000,
600000, 900000])
ax.set_xticklabels([0,
'3e5','6e5','9e5'])


## Set y (deformation)
labels y_max =
np.max(results[0][1][1])
y_min =
np.min(results[0][1][1])
ax.set_ylabel('Deformation
(m)', rotation = 100);
ax.set_yticks(np.linspace(y
_min, y_max, 4))
ax.yaxis.set_major_formatter(mtick.FormatStrFormat
ter('%.e'))

## Set z (safety factor) labels z_max =
np.max(results[0][1][2]) z_min =
np.min(results[0][1][2])
ax.set_zticks(np.linspace(z_min, z_max,
4))
ax.zaxis.set_major_formatter(mtick.Form
atStrFormatter('%.1f'))
ax.set_zlabel('Safety Factor');

## Peaks of y (deformation) plotted in
orange peaks_y, _ =
find_peaks(results[0][1][1])
ax.plot(results[0][1][0][peaks_y],
results[0][1][1][peaks_y],␣
↪results[0][1][2][peaks_y], "x")

## Peaks of z (safety factor) plotted
in green peaks_z, _ =
find_peaks(results[0][1][2])
ax.plot(results[0][1][0][peaks_z],
results[0][1][1][peaks_z],␣
↪results[0][1][2][peaks_z], "x")

## Valleys of y (deformation) plotted in
red valleys_y, _ =
find_peaks(results[0][1][1] * -1)
ax.plot(results[0][1][0][valleys_y],
results[0][1][1][valleys_y],␣
↪results[0][1][2][valleys_y], "o")

## Valleys of z (safety factor)
plotted in purple valleys_z, _ =
find_peaks(results[0][1][2] * -1)
```

```
[14]: ### Save maximum and minimum values:
     finder['peaks_df_y'] =
     list(zip(results[0][1][0][peaks_y],␣
      ↪results[0][1][1][peaks_y],
     results[0][1][2][peaks_y])) finder['peaks_sf_z'] =
     list(zip(results[0][1][0][peaks_z],␣
      ↪results[0][1][1][peaks_z],
     results[0][1][2][peaks_z])) finder['valleys_df_y']
     = list(zip(results[0][1][0][valleys_y],␣
      ↪results[0][1][1][valleys_y],
     results[0][1][2][valleys_y]))
     finder['valleys_sf_z'] =
     list(zip(results[0][1][0][valleys_z],␣
      ↪results[0][1][1][valleys_z],
      results[0][1][2][valleys_z]))
```

```
[15]: ### Output saved values (maxima and minima) to csv for
3d maxima/minima
```

```
## Create output filename for maximum of
deformations in 3D (on y-axis) f = open('deliv/' +
'extrema_3D/' + str(filename) +
"_3D_deform_max_y.csv",␣ ↪'w+')

## Write to file
f.write("Node
Number\tDeformation
(m)\n") output =
finder['peaks_df_y']
for i in
range(len(output)):
```

Converting Datasets

The code below reads the .txt files of the datasets from the ’data’ directory and saves them as a csv file in the ’data-csv’ directory for future use.

```
[76]: import os
      import pandas as pd
      from tqdm import tqdm

      # Load and Save directories
      load_directory = 'data' # contains .txt dataaset files
      save_directory = 'data-csv' # folder to save .csv dataset files

      for filename in tqdm(os.listdir(load_directory)):
          if filename.endswith(".txt"):

              print(filename)

              # Load the file
              path = os.path.join(load_directory, filename)
              df = pd.read_csv(path, sep='\t', engine='python')

              # Save the file with .csv extension
              filename = filename[:-4] + '.csv'
              save_path = os.path.join(save_directory, filename)
      #         df.to_csv(save_path, index=None)
```

Exploratory Data Analysis

Load Standard Libraries for Data Handling and Visualization

```
[6]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from tqdm import tqdm

     plt.style.use('seaborn')
     import warnings
     warnings.filterwarnings('ignore')
     %matplotlib inline
```

Load Data

We will only consider deformation data, since this is a proof of concept kind of exercise.

If a solution is found, then the stress data can also be used in a similar fashion.

```
[7]: P1_2500_deformation = pd.read_csv('data-csv/P1_2500_deformation.csv')
     P1_5000_deformation = pd.read_csv('data-csv/P1_5000_deformation.csv')
     P1_7500_deformation = pd.read_csv('data-csv/P1_7500_deformation.csv')
     P1_10000_deformation = pd.read_csv('data-
     csv/P1_10000_deformation.csv')



     P2_2500_deformation = pd.read_csv('data-csv/P2_2500_deformation.csv')
     P2_5000_deformation = pd.read_csv('data-csv/P2_5000_deformation.csv')
     P2_7500_deformation = pd.read_csv('data-csv/P2_7500_deformation.csv')
     P2_10000_deformation = pd.read_csv('data-csv/P2_10000_deformation.csv')

     P3_2500_deformation = pd.read_csv('data-csv/P3_2500_deformation.csv')
     P3_5000_deformation = pd.read_csv('data-csv/P3_5000_deformation.csv')
     P3_7500_deformation = pd.read_csv('data-csv/P3_7500_deformation.csv')
     P3_10000_deformation = pd.read_csv('data-csv/P3_10000_deformation.csv')

     P4_2500_deformation = pd.read_csv('data-csv/P4_2500_deformation.csv')
     P4_5000_deformation = pd.read_csv('data-csv/P4_5000_deformation.csv')
```

```
P4_7500_deformation = pd.read_csv('data-csv/P4_7500_deformation.csv')
P4_10000_deformation = pd.read_csv('data-csv/P4_10000_deformation.csv')

# Check one
P1_2500_deformation.head()
```

[7]:

```
     Node Number Total Deformation (m)
0              1  0.000181
1              2  0.000193
2              3  0.000182
3              4  0.000182
4              5  0.000183
```

[3]:

```
print(P1_2500_deformation.isnull().sum())
print(P1_5000_deformation.isnull().sum())
print(P1_7500_deformation.isnull().sum())
print(P1_10000_deformation.isnull().sum())

print(P2_2500_deformation.isnull().sum())
print(P2_5000_deformation.isnull().sum())
print(P2_7500_deformation.isnull().sum())
print(P2_10000_deformation.isnull().sum())

print(P3_2500_deformation.isnull().sum())
print(P3_5000_deformation.isnull().sum())
print(P3_7500_deformation.isnull().sum())
print(P3_10000_deformation.isnull().sum())

print(P4_2500_deformation.isnull().sum())
print(P4_5000_deformation.isnull().sum())
print(P4_7500_deformation.isnull().sum())
print(P4_10000_deformation.isnull().sum())
```

```
Node Number                    0

Total Deformation (m) dtype:   0

int64

Node Number                    0
```

```
Total Deformation (m) dtype:    0
int64
Node Number                     0
Total Deformation (m) dtype:    0
int64
Node Number                     0
Total Deformation (m)           0
dtype: int64
Node Number                     0
Total Deformation (m) dtype:    0
int64
Node Number                     0
Total Deformation (m) dtype:    0
int64
Node Number                     0
Total Deformation (m) dtype:    0
int64
Node Number                     0
Total Deformation (m) dtype:    0
int64
Node Number                     0
Total Deformation (m) dtype:    0
int64
```

```
Node Number                  0
Total Deformation (m) dtype: 0
int64
Node Number                  0
Total Deformation (m) dtype: 0
int64
Node Number                  0
Total Deformation (m) dtype: 0
int64
Node Number                  0
Total Deformation (m) dtype: 0
int64
Node Number                  0
Total Deformation (m) dtype: 0
int64
Node Number                  0
Total Deformation (m) dtype: 0
int64
Node Number                  0
Total Deformation (m) dtype: 0
int64
```

The data does not contain any NaN or null values.

Creating Image Data for Each Class

The following code will bin the “Node Number” into “Node Number Binned” column, which will later be used to create graph images for each class.

```
[8]: binwidth = int(max(P1_2500_deformation["Node Number"]) - ␣
      ↪min(P1_2500_deformation["Node Number"]))/7 bins
     = range(int(min(P1_2500_deformation["Node
     Number"])), ␣
      ↪int(max(P1_2500_deformation["Node Number"])), int(binwidth))
     group_names = ["1/6", "2/6", "3/6", "4/6", "5/6", "6/6", ]


     P1_2500_deformation["Node Number Binned"] =
     pd.cut(P1_2500_deformation["Node ␣
      ↪Number"], bins, labels=group_names)
     P1_5000_deformation["Node Number Binned"] =
     pd.cut(P1_5000_deformation["Node ␣
      ↪Number"], bins, labels=group_names)
     P1_7500_deformation["Node Number Binned"] =
     pd.cut(P1_7500_deformation["Node ␣
      ↪Number"], bins, labels=group_names)
     P1_10000_deformation["Node Number Binned"] =
       pd.cut(P1_10000_deformation["Node ␣ ↪Number"], bins, labels=group_names)


     P2_2500_deformation["Node Number Binned"] =
     pd.cut(P2_2500_deformation["Node ␣
      ↪Number"], bins, labels=group_names)
     P2_5000_deformation["Node Number Binned"] =
     pd.cut(P2_5000_deformation["Node ␣
      ↪Number"], bins, labels=group_names)
```

```
P2_7500_deformation["Node Number Binned"] =
pd.cut(P2_7500_deformation["Node␣
  ↪Number"], bins, labels=group_names)

P2_10000_deformation["Node Number Binned"] =
  pd.cut(P2_10000_deformation["Node␣ ↪Number"], bins, labels=group_names)


P3_2500_deformation["Node Number Binned"] =
pd.cut(P3_2500_deformation["Node␣
  ↪Number"], bins, labels=group_names)

P3_5000_deformation["Node Number Binned"] =
pd.cut(P3_5000_deformation["Node␣
  ↪Number"], bins, labels=group_names)

P3_7500_deformation["Node Number Binned"] =
pd.cut(P3_7500_deformation["Node␣
  ↪Number"], bins, labels=group_names)

P3_10000_deformation["Node Number Binned"] =
  pd.cut(P3_10000_deformation["Node␣ ↪Number"], bins, labels=group_names)


P4_2500_deformation["Node Number Binned"] =
pd.cut(P4_2500_deformation["Node␣
  ↪Number"], bins, labels=group_names)

P4_5000_deformation["Node Number Binned"] =
pd.cut(P4_5000_deformation["Node␣
  ↪Number"], bins, labels=group_names)

P4_7500_deformation["Node Number Binned"] =
pd.cut(P4_7500_deformation["Node␣
  ↪Number"], bins, labels=group_names)

P4_10000_deformation["Node Number Binned"] =
  pd.cut(P4_10000_deformation["Node␣ ↪Number"], bins, labels=group_names)


P1_2500_deformation.dropna(axis=0, inplace=True)
P1_5000_deformation.dropna(axis=0, inplace=True)
P1_7500_deformation.dropna(axis=0, inplace=True)
P1_10000_deformation.dropna(axis=0, inplace=True)


P2_2500_deformation.dropna(axis=0, inplace=True)
```

```
P2_5000_deformation.dropna(axis=0, inplace=True)
P2_7500_deformation.dropna(axis=0, inplace=True)
P2_10000_deformation.dropna(axis=0, inplace=True)

P3_2500_deformation.dropna(axis=0, inplace=True)
P3_5000_deformation.dropna(axis=0, inplace=True)
P3_7500_deformation.dropna(axis=0, inplace=True)
P3_10000_deformation.dropna(axis=0, inplace=True)

P4_2500_deformation.dropna(axis=0, inplace=True)
P4_5000_deformation.dropna(axis=0, inplace=True)
P4_7500_deformation.dropna(axis=0, inplace=True)
P4_10000_deformation.dropna(axis=0, inplace=True)
```

Plot Graphs of the Classes

```
[9]: # Save function is commented, as the code has already run and images created in␣
     ↪the director
     # Remove the tile tag when recreating images, and uncomment the savefig lines

     ax1 = sns.relplot("Node Number Binned", "Total Deformation (m)",␣
     ↪data=P1_2500_deformation, kind='line') ax1.set(ylim=(0,
     0.012), title="P1 2500 deformation") # ax1.savefig("class-
     images/P1_2500_deformation.png") ax2 = sns.relplot("Node
     Number Binned", "Total Deformation (m)",␣
     ↪data=P1_5000_deformation, kind='line') ax2.set(ylim=(0,
     0.012), title="P1 5000 deformation") # ax2.savefig("class-
     images/P1_5000_deformation.png") ax3 = sns.relplot("Node
     Number Binned", "Total Deformation (m)",␣
     ↪data=P1_7500_deformation, kind='line') ax3.set(ylim=(0,
     0.012), title="P1 7500 deformation") # ax3.savefig("class-
     images/P1_7500_deformation.png") ax4 = sns.relplot("Node
     Number Binned", "Total Deformation (m)",␣
     ↪data=P1_10000_deformation, kind='line')
     ax4.set(ylim=(0, 0.012), title="P1 10000
     deformation") # ax4.savefig("class-
     images/P1_10000_deformation.png")

     ax5 = sns.relplot("Node Number Binned", "Total Deformation (m)",␣
     ↪data=P2_2500_deformation, kind='line') ax5.set(ylim=(0,
     0.012), title="P2 2500 deformation") # ax5.savefig("class-
     images/P2_2500_deformation.png") ax6 = sns.relplot("Node
     Number Binned", "Total Deformation (m)",␣
     ↪data=P2_5000_deformation, kind='line') ax6.set(ylim=(0,
     0.012), title="P2 5000 deformation") # ax6.savefig("class-
     images/P2_5000_deformation.png") ax7 = sns.relplot("Node
     Number Binned", "Total Deformation (m)",␣
     ↪data=P2_7500_deformation, kind='line')
```

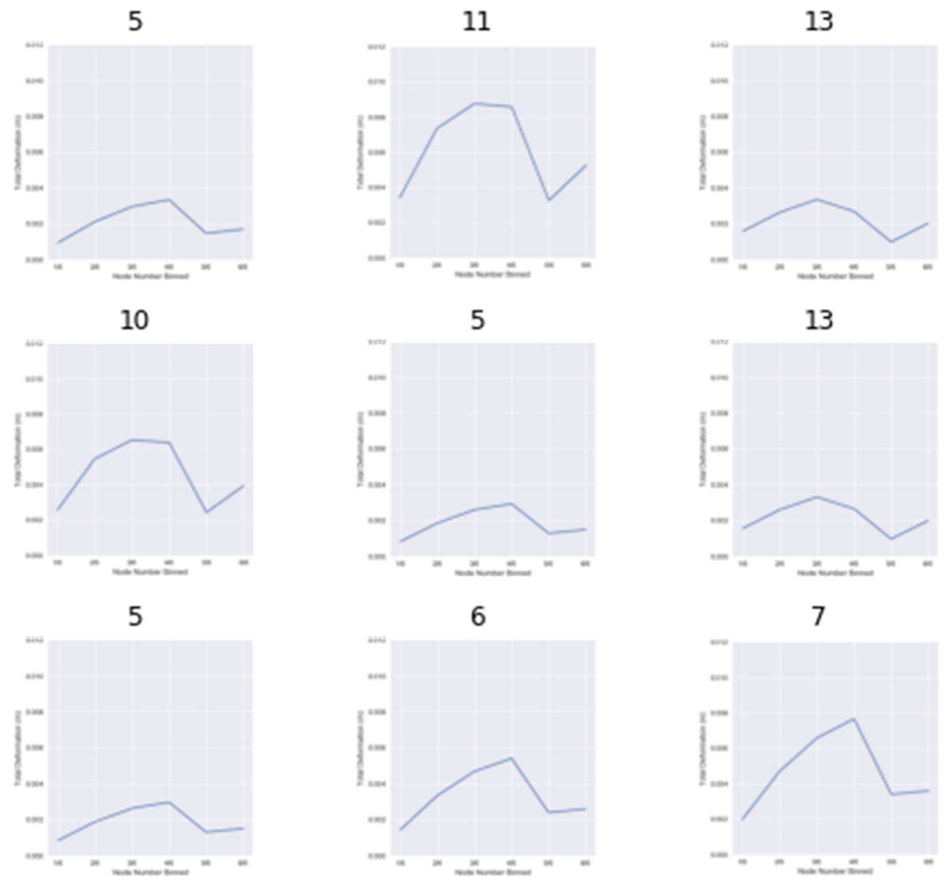
Training

Fastai library is used to train a neural network model using the augmented images created from the “EDA and Data Preparation” notebook.

```
[1]:  from fastai.vision import *
      from fastai.metrics import error_rate
```

A DataBunch is created using the saved augmented images in the ‘augmented_images’ directory (in my case, Google Drive). The size of the image is 347px, while 20% of the images are used for validation.

```
[2]: data = ImageDataBunch.from_folder('/content/drive/My Drive/Upwork/Comparison
       ␣ ↪Neural Network/augmented_images', size=347, valid_pct=0.2).
      ↪normalize(imagenet_stats) data.show_batch(rows=3, figsize=(7,6))
```



```
[2]: # DataBunch is aware of our 16 class targets.
     print(data.classes)
     len(data.classes),data.c
```

```
[4]: # Create a model using resnet34 architecture
     learn = cnn_learner(data, models.resnet34, metrics=error_rate)
```

```
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to

/root/.cache/torch/checkpoints/resnet34-333f7ec4.pth
```

```
HBox(children=(FloatProgress(value=0.0, max=87306240.0), HTML(value='')))
```

```
[5]: # Description of the model
     learn.model
```

```
[5]: Sequential(
  (0): Sequential(
    (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
    (4): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)

        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

      )

      (1): BasicBlock(

        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)

        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

        (relu): ReLU(inplace=True)

        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)

        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

      )

      (2): BasicBlock(

        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
```

```
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

        (relu): ReLU(inplace=True)

        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)

        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

      )

    )

    (5): Sequential(

      (0): BasicBlock(

        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)

        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

        (relu): ReLU(inplace=True)

        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
```

```
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      )
      (2): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
      (3): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
```

```
            (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
        )
        (6): Sequential(
          (0): BasicBlock(
            (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
            (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (relu): ReLU(inplace=True)
            (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (downsample): Sequential(
               (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
             (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            )
          )
          (1): BasicBlock(
            (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (relu): ReLU(inplace=True)
            (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          )
          (2): BasicBlock(
            (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

            (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

            (relu): ReLU(inplace=True)

            (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

            (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

          )

          (3): BasicBlock(

            (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

            (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)

            (relu): ReLU(inplace=True)

            (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
```

```
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

      )

      (4): BasicBlock(

        (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

        (relu): ReLU(inplace=True)

        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

      )

      (5): BasicBlock(

        (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
        (relu): ReLU(inplace=True)

        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

      )

    )

    (7): Sequential(

      (0): BasicBlock(

        (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)

        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

        (relu): ReLU(inplace=True)

        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
      (2): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

        (relu): ReLU(inplace=True)

        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)

        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)

      )

    )

  )

  (1): Sequential(

    (0): AdaptiveConcatPool2d(

      (ap): AdaptiveAvgPool2d(output_size=1)

      (mp): AdaptiveMaxPool2d(output_size=1)

    )

    (1): Flatten()
```

```
        (2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (3): Dropout(p=0.25, inplace=False)
        (4): Linear(in_features=1024, out_features=512, bias=True)
        (5): ReLU(inplace=True)
        (6): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (7): Dropout(p=0.5, inplace=False)
        (8): Linear(in_features=512, out_features=16,
    bias=True) )

)
```

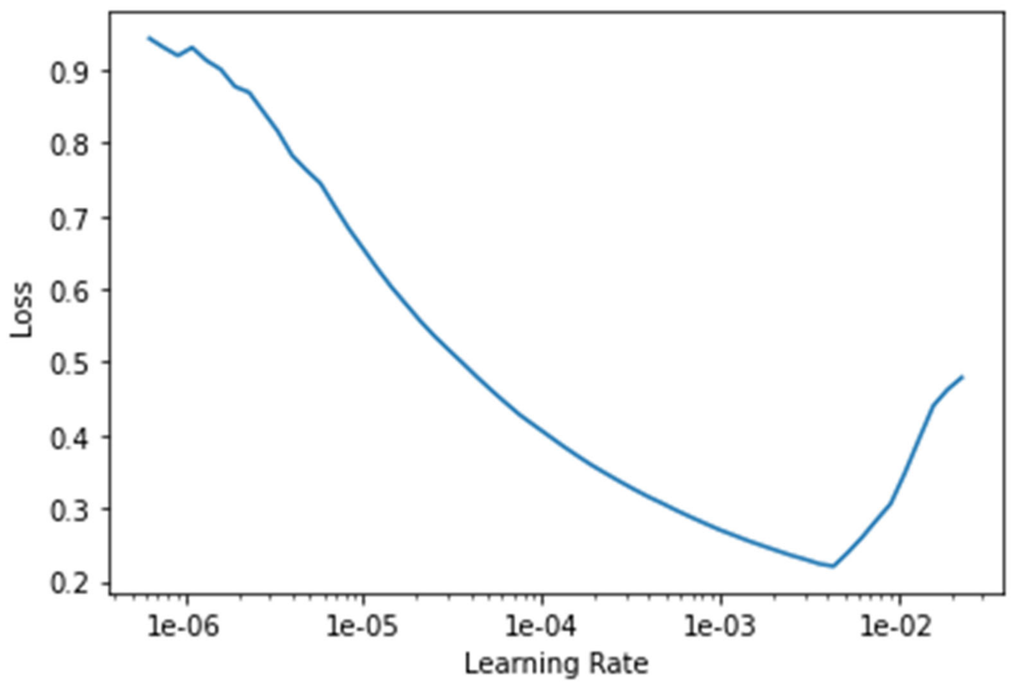
```
[6]: # Training the model using 4 epochs
     learn.fit_one_cycle(4)
```

With the default layers and restnet 34, the error rate is very high. Let us unfreeze and study the learning rate.

```
[7]: learn.unfreeze()
```

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

[12]:
```
learn.recorder.plot()
```



[4]:
```
# Using the best learning rate slice
learn.unfreeze()
learn.fit_one_cycle(2, max_lr=slice(1e-6,1e-4))
```

0.1.1 Next, we use Restnet50 and a better learning rate to gain improved accuracy.

[14]:
```
data = ImageDataBunch.from_folder('/content/drive/My Drive/Upwork/Comparison
 ↪Neural Network/augmented_images', size=347, valid_pct=0.2).
↪normalize(imagenet_stats)
```

[15]:
```
learn = cnn_learner(data, models.resnet50, metrics=error_rate)
```

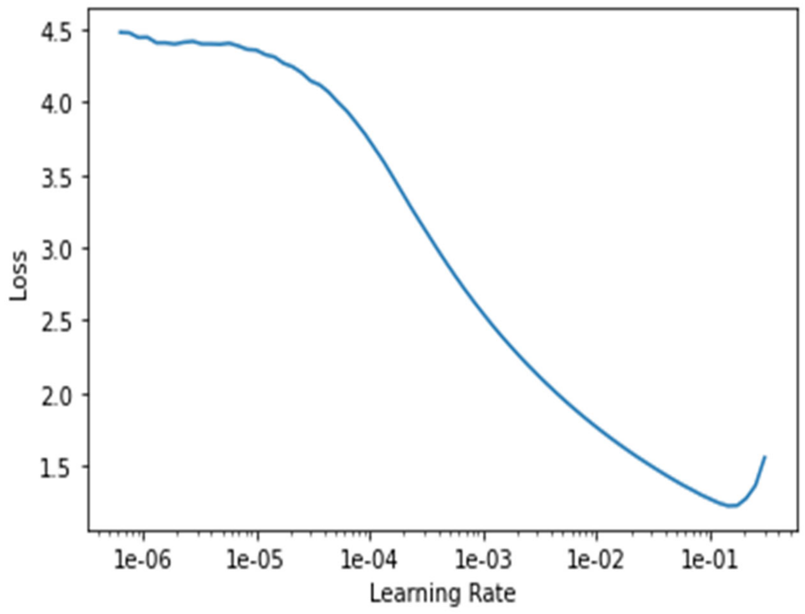
```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to

/root/.cache/torch/checkpoints/resnet50-19c8e357.pth

HBox(children=(FloatProgress(value=0.0, max=102502400.0), HTML(value='')))
```

```
[16]: learn.lr_find()
      learn.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the
```



```
[17]: learn.fit_one_cycle(8)
```

```
[ ]: learn.save('stage-1-50')
```

```
[18]: learn.unfreeze()
      learn.fit_one_cycle(3, max_lr=slice(1e-6,1e-4))
```