Department of Mechanical Engineering

FACULTY OF ENGINEERING AND DESIGN

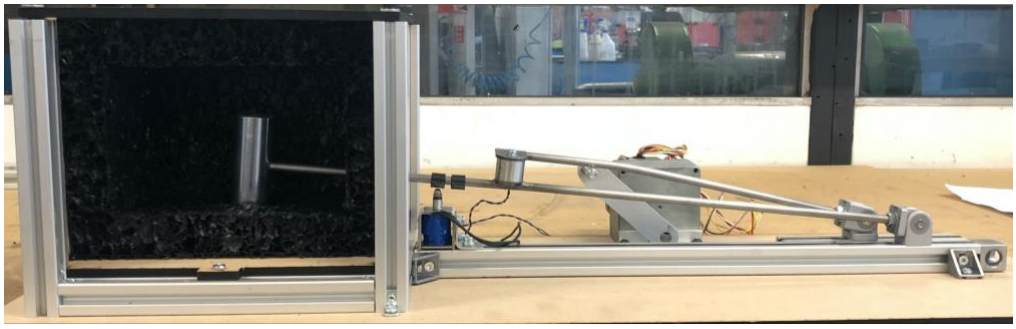
FINAL YEAR MEng PROJECT REPORT

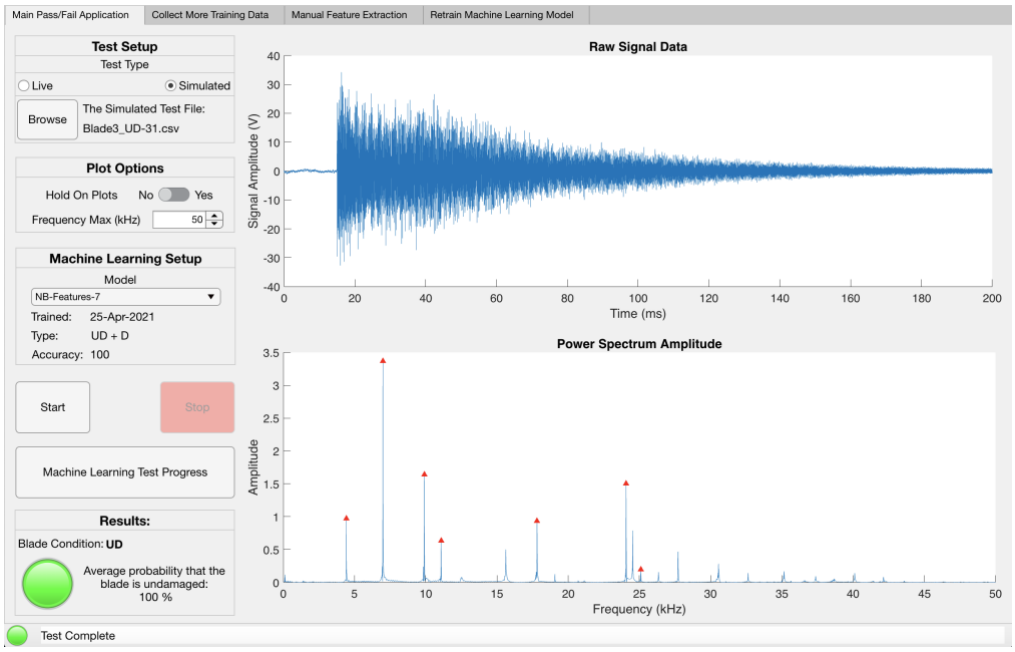
Pass/Fail Testing of Additive Manufacturing Structures Using Machine Learning Algorithms

Nicholas Thomas

28th April 2021

Word count: 12,977





"I certify that I have read and understood the entry in the Student Handbook for the Department of Mechanical Engineering on Cheating and Plagiarism and that all material in this assignment is my own work, except where I have indicated with appropriate references."

Author's signature: 

Supervisor: *Prof Michele Meo*

Assessor: *Dr James Scobie*

Summary

The use of additively manufactured metal components is growing rapidly in the aerospace industry where fuel consumption and costs drive the need for lighter and more complex parts. Additively manufactured parts have metallurgic differences compared to machined parts, for example mechanical anisotropy, residual stress and defects. These defects, also known as damage, reduce the mechanical performance of the part and hence are a source for concern when designing safety critical components. Current methods to inspect additively manufactured parts for defects include industrial micro computed-tomography imaging and extensive destructive testing, which are time consuming and expensive.

This project focused on developing a quick, reliable vibration testing method with machine learning algorithms to discern an undamaged additively manufactured turbine blade from a damaged one. This has been achieved by designing and manufacturing a test rig alongside developing a MATLAB graphical user interface.

The test rig was developed using the systematic engineering design process. The test rig used the impulse excitation technique, a non-destructive testing method, to produce the vibration response. The vibration response was passed through a Fast Fourier Transform to identify the natural frequencies which change value depending on the quantity of defects present and therefore the damage. The graphic user interface was created to classify a turbine blade, to collect training data for the machine learning model, to extract the key features, and to create and train the various machine learning models using the identified natural frequencies. The graphical user interface offered the opportunity to train several different algorithms for supervised machine learning models, namely binary classification tree, k-nearest neighbours, naïve Bayes, discriminant analysis, shallow neural network and deep learning, and unsupervised clustering models such as density-based spatial clustering of applications with noise and Gaussian mixture.

Due to the Covid-19 restrictions, a limited amount of analysis was conducted on the performance of the various machine learning models, as only a small amount of test data was available, however, key themes were identified. Conventional supervised machine learning models took 38 seconds to train on average, compared to 7 minutes for deep learning models. The trained accuracy of all these models was 100% when classifying known blades. Conversely, when classifying unknown blades, the deep learning models were inconsistent, whilst all the conventional machine learning models were accurate. The various unsupervised models were also all accurate when classifying known blades. It was identified that the selection of natural frequency peaks was directly related to the accuracy of the machine learning model.

Overall, the test rig and MATLAB application were deemed successful to classify the damage condition of turbine blades but would need to be fully validated with a comprehensive dataset of blades.

Acknowledgements

I would first like to thank my project supervisor, Professor Michele Meo who has been very encouraging and enthusiastic; helping me to push my boundaries to produce a useful test rig and software application which I hope will be valuable to him and his team. I came into the project with little to no knowledge of many non-destructive testing methods, machine learning algorithms or additive manufacturing limitations and I have come out the other side invested in the future of artificial intelligence.

I would like to thank my assessor, Dr James Scobie who has given me excellent constructive advice during our meetings.

I am extremely grateful for the support and time Marco Boccaccio has given me with collecting the test data, providing advice with the design of the new test rig compared to the old one and generally supporting me through the restrictions.

I would also to thank all the technicians who helped with the manufacture of the test rig due to the Covid-19 restrictions, especially Dave Wood, Jeff Boston, John Bishop and Guy Brace.

Finally, I would like to thank the support of my family throughout my degree particularly during the Covid-19 pandemic. Special thanks go to my mum who has been incredibly supportive throughout my whole degree, especially during this project.

Table of Contents

List of Abbreviations

AE	Acoustic Emission
AI	Artificial Intelligence
AM	Additive Manufacturing
CAD	Computer-Aided Design
CMM	Coordinate-Measuring Machine
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CSV	Comma-Separated Values
CT	Computed Tomography
DFT	Discrete Fourier Transform
DL	Deep Learning
DT	Destructive Testing
ET	Electromagnetic Testing
FEA	Finite Element Analysis
FFT	Fast Fourier Transform
Gage R&R	Gage Repeatability & Reproducibility
GPE	Gravitational Potential Energy
GPU	Graphics Processing Unit
GUI	Graphical User Interface
IET	Impulse Excitation Technique
MDF	Medium Density Fibreboard
ML	Machine Learning
NDT	Non-Destructive Testing
PCA	Principle Component Analysis
PCRT	Process Compensated Resonance Testing
PT	Penetration Testing
RUS	Resonant Ultrasound Spectroscopy
3D	Three-Dimensional
U	Elastic Strain Energy
UT	Ultrasonic Testing
WD	Work Done

List of Symbols

d	Distance
f	Frequency
F	Force
g	Acceleration of gravity
h	Height above datum
k	Stiffness
m	Mass
$P(p_1, p_2)$	X and Y coordinate of point P
$Q(p_1, p_2)$	X and Y coordinate of point Q
t	Thickness
v	Speed of sound
V	Volume of the object
ε	Strain
λ	Wavelength
π	Pi
σ	Stress

List of Figures

List of Tables

1 Introduction

1.1 Background

Additive Manufacturing (AM) is a type of manufacturing process which was developed in the 1980s. It has advanced at an extraordinary pace, to the point where today, anyone at home can print their own designs enabling a quick development process [1]. In fact, it is described as an essential technology of the 4th Industrial revolution [2], [3]. In AM, material is added to create an object instead of using the traditional subtractive manufacturing process. The process builds Three-Dimensional (3D) parts directly from a Computer-Aided Design (CAD) model, usually by depositing successive thin layers of material, which are normally plastic or metal, on top of each other. One of these methods, namely laser sintering, is shown in Figure 1 [4].

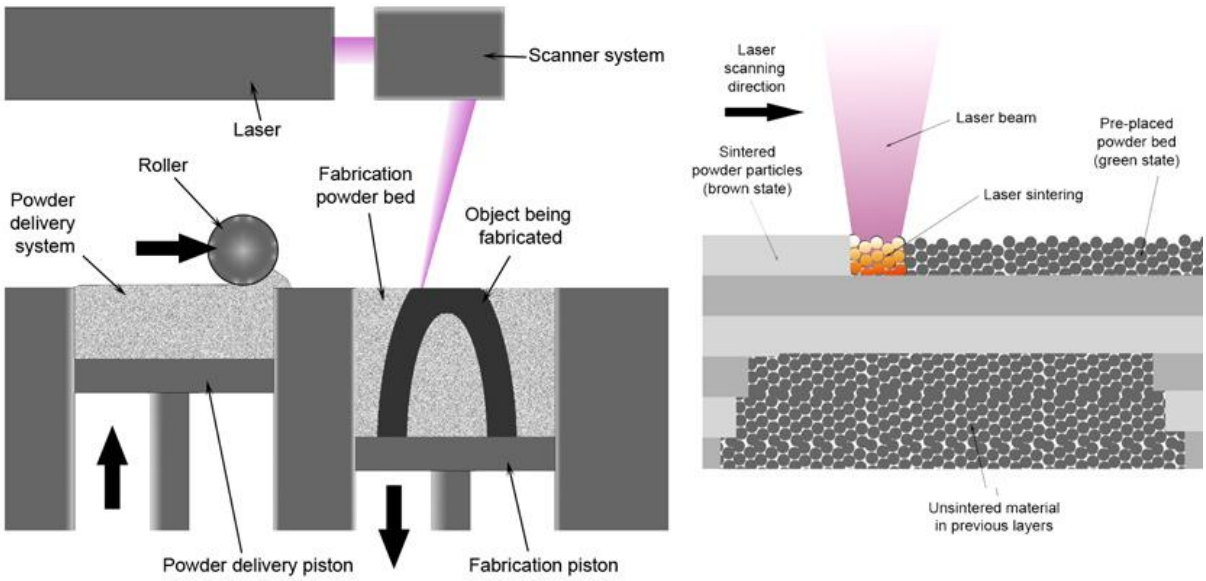


Figure 1 – Diagram of the laser sintering method [5]

AM solves problems associated with traditional subtractive processes. Firstly, it allows unique designs to be created, such as producing parts with internal support structures to reduce weight, which would have been impossible using subtractive manufacturing methods. Moreover, the material efficiency of the manufacturing process is greatly improved, as only the required material is used instead of subtracting material from a large starting block [6]. This has advantages; for instance, it reduces the environmental impact of shipping the raw material and offcuts. Furthermore, there is a reduced impact of the toxic by-products found in subtractive machines, on the air, water and soil [7]. Moreover, designs can be manufactured without the need for expensive tooling or forms, which ultimately saves cost [8].

AM is revolutionising the manufacturing industry, in particular the aerospace sector, where parts need to be lighter and more complex to save fuel. Figure 2 presents a selection of AM parts of an aircraft engine and Figure 3 shows the design optimisation possibilities for a simple component of an Airbus plane. AM has also been proved to be a viable manufacturing solution in space, on board the International Space Station [9].

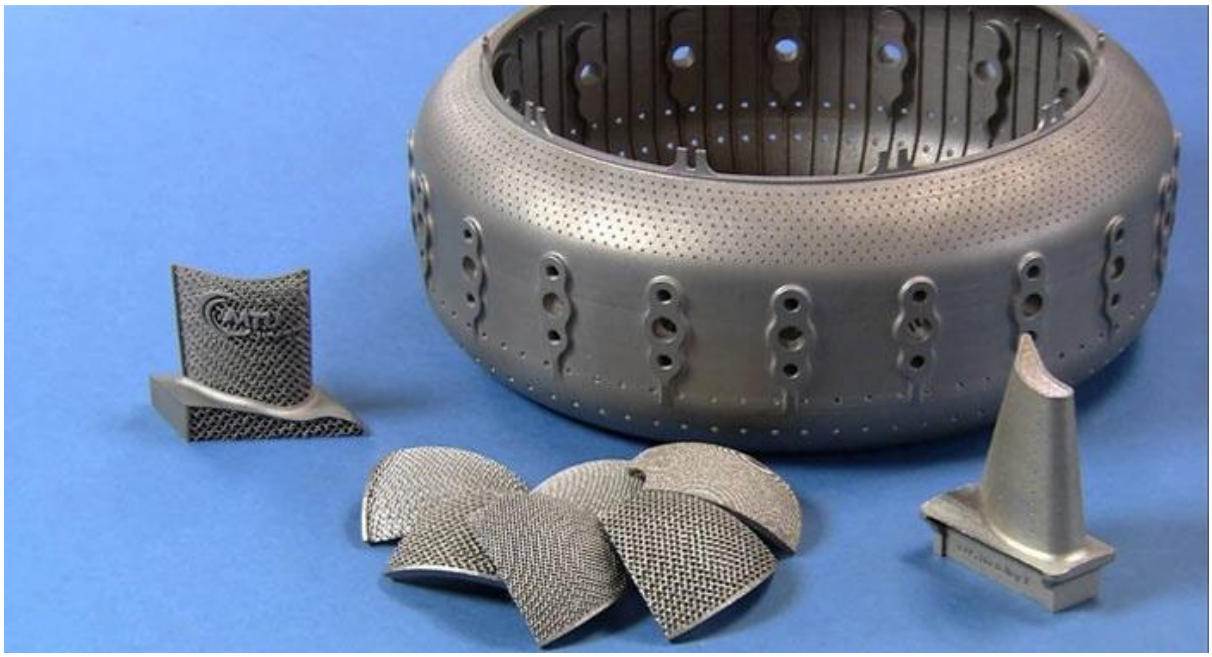


Figure 2 – Examples of AM parts for an aero engine [10]

Figure 3 – Comparison of a machined versus an additively manufactured part [10]

However, problems can emerge with components manufactured by metal AM, as they can be affected by several types of defects. These defects have a negative impact on the functional performance of the part and decrease confidence when designing components using this manufacturing method, particularly if they are safety critical [11]. The effect of these defects on the component can be difficult to quantify. Current commercial methods are based on extensive destructive testing and industrial micro computed-tomography imaging, as seen in Figure 4. These are both time consuming and expensive methods [11].

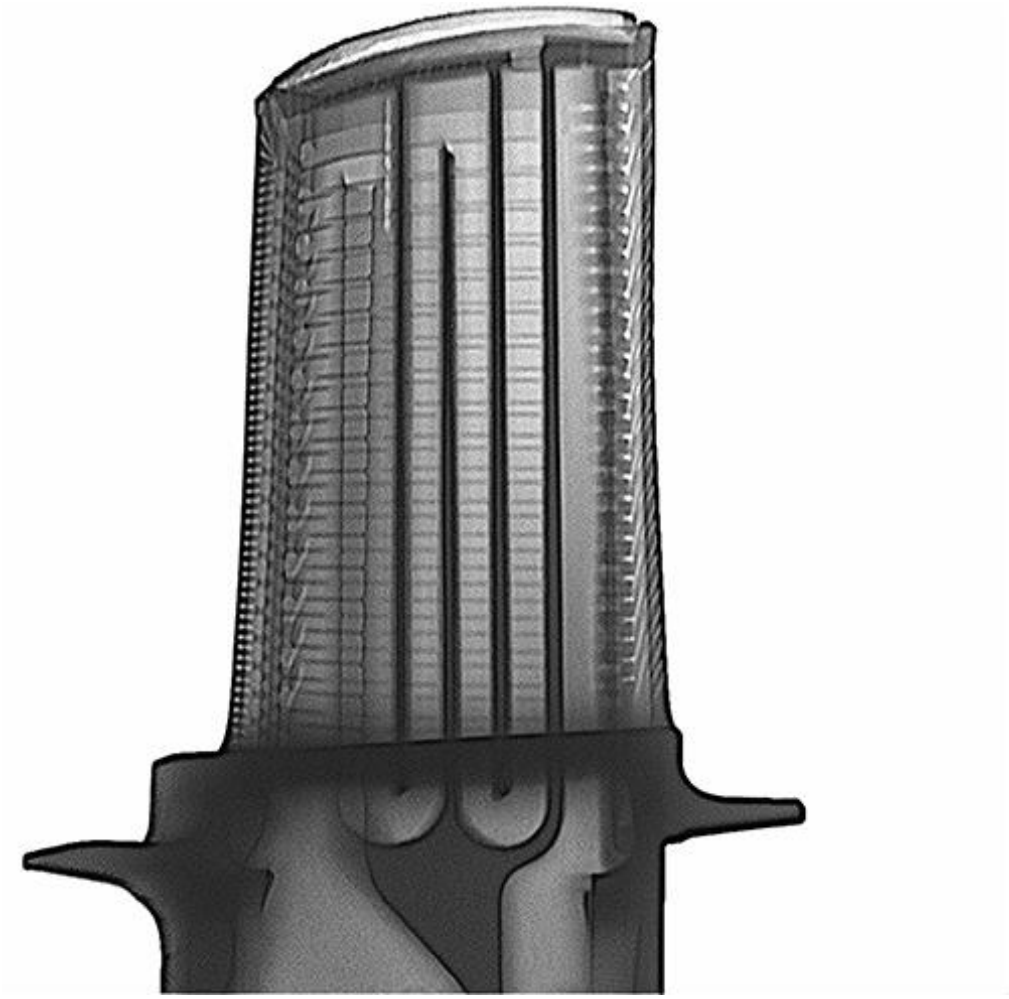


Figure 4 – A computed-tomography image of a turbine blade [12]

Artificial Intelligence (AI) and therefore Machine Learning (ML), a subset of AI, is also described to be another essential technology of the 4th Industrial revolution [2]. AI allows computers to mimic the human mind, where it learns from experience, can recognise objects, make decisions, and solve complex problems. AI has become extremely popular in recent years due to the advancements in Central Processing Unit (CPU) and Graphics Processing Unit (GPU) both producing more computing power, the extensive quantity of data and the improvements in algorithms [13]. These coupled together allow AI to extract unique patterns from large datasets.

1.2 Aims and Objectives

The aim of the project was to develop a quick, reliable vibration testing method with machine learning algorithms to discern an undamaged additively manufactured turbine blade from a damaged one.

Present methods are time consuming and expensive, leading to a desire for a quick, reliable and cost-effective system for quantifying the effect of defects on AM turbine blades. Research highlighted the possibility of using ML to conduct the analysis, as humans are unable to do it effectively due to the large amount of data involved. The aim was achieved by completing two stages: a mechanical development stage and a software development stage, both of which contained objectives as summarised below:

1. Create a test rig and corresponding test method to collect vibration test data
    a. Design a test rig to hold the blades during repeated testing without affecting the results
    b. Create a test procedure to reliably extract vibration data from AM blades using the test rig
    c. Manufacture and construct the test rig, integrating the sensors
2. Create a ML algorithm to discern an undamaged blade from a damaged one
    a. Process the test data to remove the noise and extract key features from the test signals
    b. Develop supervised ML algorithms in software
    c. Train the algorithms with the training dataset of undamaged and damaged blades
    d. Create a GUI to display the results, add to the training dataset, perform manual feature extraction, and train the ML algorithm
    e. Automate the test process and result generation by interfacing directly with the hardware
    f. Validate the supervised ML algorithms against an unseen blade of known condition
    g. Develop unsupervised ML algorithms
    h. Train the algorithms with the training dataset of undamaged blades only

A test rig and accompanying Graphical User Interface (GUI) capable of classifying the damage condition of AM objects were the outcomes of the project. A mechanical test rig was required to test the blades in order to obtain the vibration response. A GUI was also necessary to process the response from the blades to create, train, and then use ML algorithms to classify new blades.

The project had to be adapted due to the Covid-19 pandemic and restrictions which prevented access to the university. As a result, the mechanical design was simplified, and a greater focus was placed on the software development of the GUI and ML algorithms. The analysis of the new test rig and ML algorithms was limited, as only a small number of blades and therefore test results were made available.

1.3 Layout of the Report

The report is structured to 'tell the story' of the project using the objectives as a logical path. The report is divided into four main sections, each one describing the methodology, key results, analysis and discussion, which relate back to the objectives.

Section 2 contains the literature review which outlines the fundamental theory of existing component test methods and ML. The outcome of the literature review will shape the test methods, techniques and algorithms that will be used.

Section 3 includes a detailed analysis of the new test rig development. This section follows the systematic engineering design process and therefore the section includes the results and discussion of the rationale behind the design.

Section 4 includes the development of the software application and ML algorithms. This section follows the software design process and therefore the section includes results and discussion of the inner workings of the GUI, as well as the selection of ML algorithms.

Section 5 contains a detailed analysis of the results and discussion from the entire test rig, which was developed in Sections 3 and 4. The experimental method and GUI are described in detail, as well as comparing the accuracy of different ML algorithms. The associated errors, uncertainty and limitations are also discussed.

2 Literature Review

The literature review researches the following topics:

- AM properties and defects
- Types of AM testing methods
- Signal processing techniques
- ML methods

These topics produce a solid understanding of the publications in the area and will shape the development of the project. This literature review is an expanded version of the one conducted for the Project Scoping and Planning report, Thomas [14].

2.1 Additively Manufactured Properties and Defects

This section investigates what defects are present in AM components and how they affect the properties, which will help achieve objective 2a. There are metallurgical differences between AM and conventionally manufactured components, such as residual stress, mechanical anisotropy and defects, as highlighted by Milewski [15]. The defects produced are porosity, cracking, inclusions, lack of fusion, trapped powder, large-scale voids, and chemical inconsistencies, Schulenburg [16]. All these defects affect the mechanical performance and material properties of a component, Meo [11].

Bartlett, et al. [17] drew attention to the fact that the defects are inherently linked to the process conditions, such as the powder bed quality; chamber temperature and gas type, scan speed and laser power.

Schulenburg [16] stated that a lack of fusion defect is generally caused when a newly deposited layer of powder is not sufficiently heated to melt. Consequently, this prevents the fusion between the new layer and the underlying solid layer. In addition, porosity occurs when the liquid metal available during solidification is not able to compensate for density changes, as the material undergoes liquid to solid phase transformation. Furthermore, voids occur when the energy level of the laser changes or an interruption in powder supply occurs. These reasons for the defects were confirmed by Mandache [18].

Figure 5 shows some of the common defects of cracks, voids, porosity, and trapped powder. A certain number of these defects can be eliminated by postprocess hot isostatic pressing, which simultaneously applies high temperature and pressure via an inert gas to a component, Tammas-Williams, et al. [19].

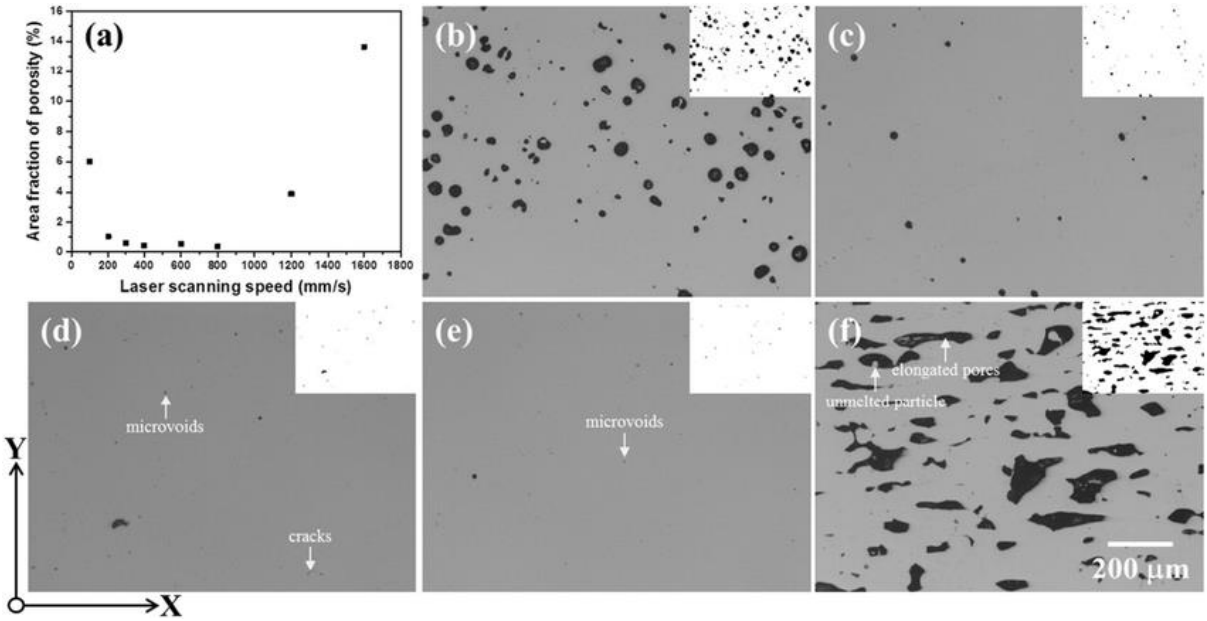


Figure 5 – Example images of common defects in AM parts [20]

Lewandowski and Seifi [21] specified that the defects, as listed, lead to a reduction in fatigue life and strength, which was confirmed by Vibrant [22]. This reduction in strength and therefore stiffness of the component is of special interest for material testing because a reduction in stiffness leads to a reduction in the natural frequency, Vibrant [22]. This change in frequency occurs for each natural frequency mode of the component. Equation 2 relates the natural frequency to the stiffness and mass of a component [23]:

$$f = \frac{1}{2\pi}\sqrt{\frac{k}{m}} \tag{2}$$

This natural frequency change can be detected by specific types of Non-Destructive Testing (NDT) methods and therefore could achieve the project aim to identify damaged and undamaged components.

2.2 Types of Additively Manufactured Testing Methods

This section investigates the different types of AM testing methods and presents the positives and negatives of each method, assisting in the selection of the most viable method to achieve objectives 1a and 1b. There are two types of testing methods used to assess components and identify the mechanical properties, namely Destructive Testing (DT) and NDT. The main characteristics and corresponding test methods of additive manufacturing are found in this British Standards (BS EN ISO 17296-3:2016) [24] document.

2.2.1 Destructive Testing Methods

DT methods simply evaluate the component to destruction. The Hobart Institute of Welding Technology [25] highlights that the experiments generally measure the strength, ductility, toughness, and hardness of the component. The most common tests performed are tensile, compression, hardness, creep, fatigue and Charpy tests. These tests would identify the properties of a single component and would be able to assess if the AM component was damaged or undamaged.

However, since DT destroys the component being tested, it was not suitable for the project, where the turbine blades were not to be damaged by the test. In addition, these methods would not achieve the aim of the project.

2.2.2 Non-Destructive Testing Methods

The alternative approach is to perform NDT on the component, which would assess if the component was damaged or undamaged, whilst keeping it intact. Mandache [18], Charalampous, et al. [26] and Kamsu-Foguem [27] present the main NDT methods that are commonly used. These are explained, citing the positives and negatives of each method in Tables 1 and 2.

The methods presented in Table 1 are not explored further as none quantify the effect the defect has on the component in terms of the mechanical properties. These methods can only identify the location, and sometimes the size of the defect, and therefore does not meet the aim of the project.

The methods presented in Table 2 are explored in greater detail as they quantify the effect of the defects on the mechanical properties and therefore achieve the aim of the project.

Table 1 – NDT methods explained and their associated positives and negatives 1

Method	Explanation	Pros	Cons
Coordinate-Measuring Machine (CMM)	• Uses a probe to inspect the surface geometry only	• Very quick	• Only tests the surface
Liquid Penetration Testing (PT)	• Uses liquid to seep into cracks and surface defects, Cartz [28]	• Very quick	• Only tests the surface
Electromagnetic Testing (ET)	• Uses a magnetic field, produced by the induced eddy currents in the sample • Types are eddy current testing and pulsed eddy current testing, Ali Sophian Guiyun Tian Mengbao [29]	• Quick	• Only tests the sub-surface and the surface defects
Industrial Computed Tomography (CT) / X-ray	• Uses irradiation to produce 3D representation of the scanned component, Carmignato, et al. [30]	• Scans the entire component	• Very slow • Expensive • Does not quantify the effect of the defects
Acoustic Emission (AE)	• Applies a force to a sample to accumulate strain energy • When released, generates elastic waves, producing an audible crack [31], Grosse and Ohtsu [32]	• Tests the entire component • Quick	• Requires an external force to be applied to the component
Ultrasonic Testing (UT) - Pulse-echo	• Uses a single ultrasonic transducer • Outputs and measures the high frequency sound waves, Tian, et al. [33]	• Tests the entire component • Quick	• Does not quantify the effect of the defects
Ultrasonic Testing (UT) - Transmission /Attenuation	• Uses two ultrasonic transducers • One acting as a pulser and the other, on the opposite side of the component, acting as a receiver, Zou, et al. [34]	• Tests the entire component • Quick	• Does not quantify the effect of the defects

Table 2 – NDT methods explained and their associated positives and negatives 2

Method	Explanation	Pros	Cons
Resonant Ultrasound Spectroscopy (RUS)	• Uses a single transmitter transducer • Uses two receiver transducers • Outputs a swept sinusoidal frequency into the component	• Tests the entire component • Very quick • Quantifies the effect of the defect	• Can only be used on small components • Multiple transducers must be used
Impulse Excitation Technique (IET)	• Uses a mechanical impulse to excite the component • Measures the induced vibration signal with a microphone or transducer	• Tests the entire component • Very quick • Quantifies the effect of the defect	• Requires an external impact

2.2.3 Review of the Resonant Ultrasound Spectroscopy Method

The RUS method evaluates the full volume of the component, rather than just the surface, with a single test as presented by Hardy, et al. [35]. This is a key advantage as only a single test is required to obtain the behaviour of the entire component, instead of the need to perform many tests at different locations, hence reducing testing time; meeting one of the main aims of the project.

Balakirev, et al. [36] explains the method where an input transducer sweeps through a frequency band of interest. As the transmitted frequency nears a natural frequency of the component, resonance occurs and generates an increased amplitude in the component. At the same time there are two receiver transducers that are measuring the frequency amplitude and can therefore identify the value of the natural frequencies of the component. An example set-up of the RUS method can be seen in Figure 6 where the output from the sensors is a resonance spectrum.

Solodov, et al. [37] stated that RUS is mainly applied to small components rather than large components where the ability to detect small defects is limited. Vibrant [38] uses the RUS method in their Process Compensated Resonance Testing (PCRT) of aerospace components. The PCRT method uses computer based analytical software with the combination of RUS to identify changes in natural frequency and their associated trends between components. Hunter [39] also comprehensively described using PCRT to assess the condition of turbine blades.

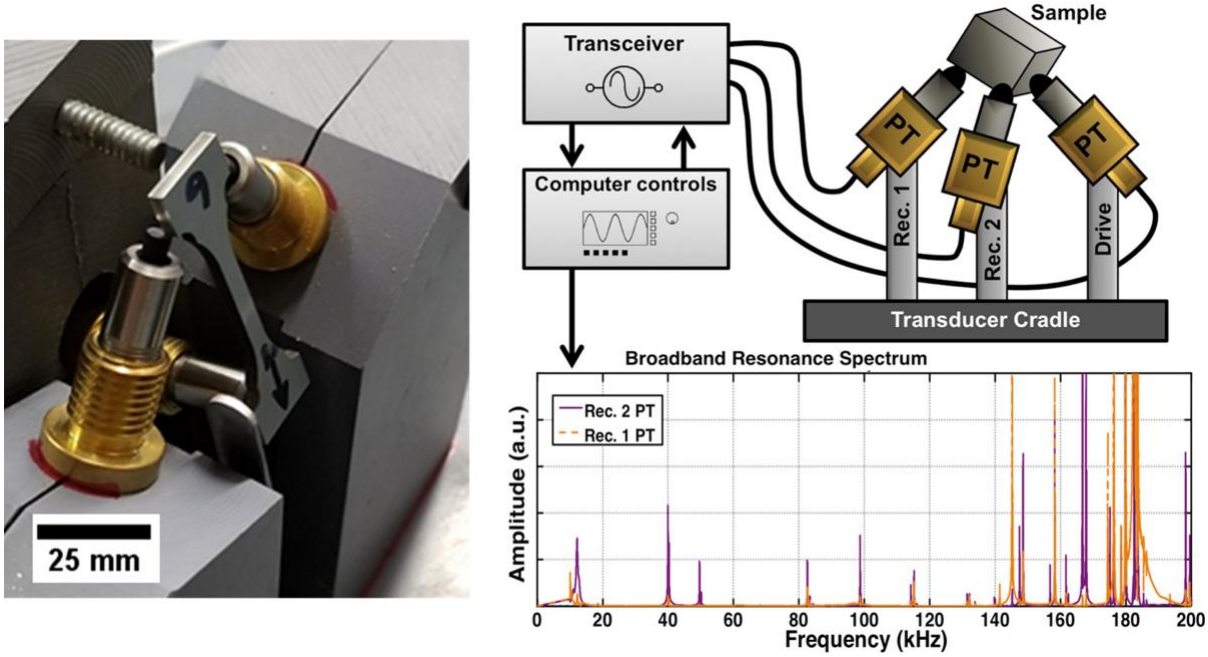


Figure 6 – Example set-up of the RUS method with a resonance spectrum output [40]

2.2.4 Review of the Impact Excitation Technique

The IET also evaluates the full volume of the component with a single test, making it a very quick method, thus fulfilling one of the main aims of the project. Psiuk, et al. [41] explained the method where an impulse excitation, normally from a hammer impact, induces vibrations inside a component. These vibrations are measured by either a receiver transducer in contact with the component, or a microphone which does not contact the component. Roebben, et al. [42] highlighted that the use of a non-contact method is preferred over a contact method because the coupling of the sensor to component affects the response. Furthermore, the repeatability is improved as the exact contact conditions change between tests for a contact transducer. An example set-up of the IET can be seen in Figure 7, where the output from the sensor is a time series signal.

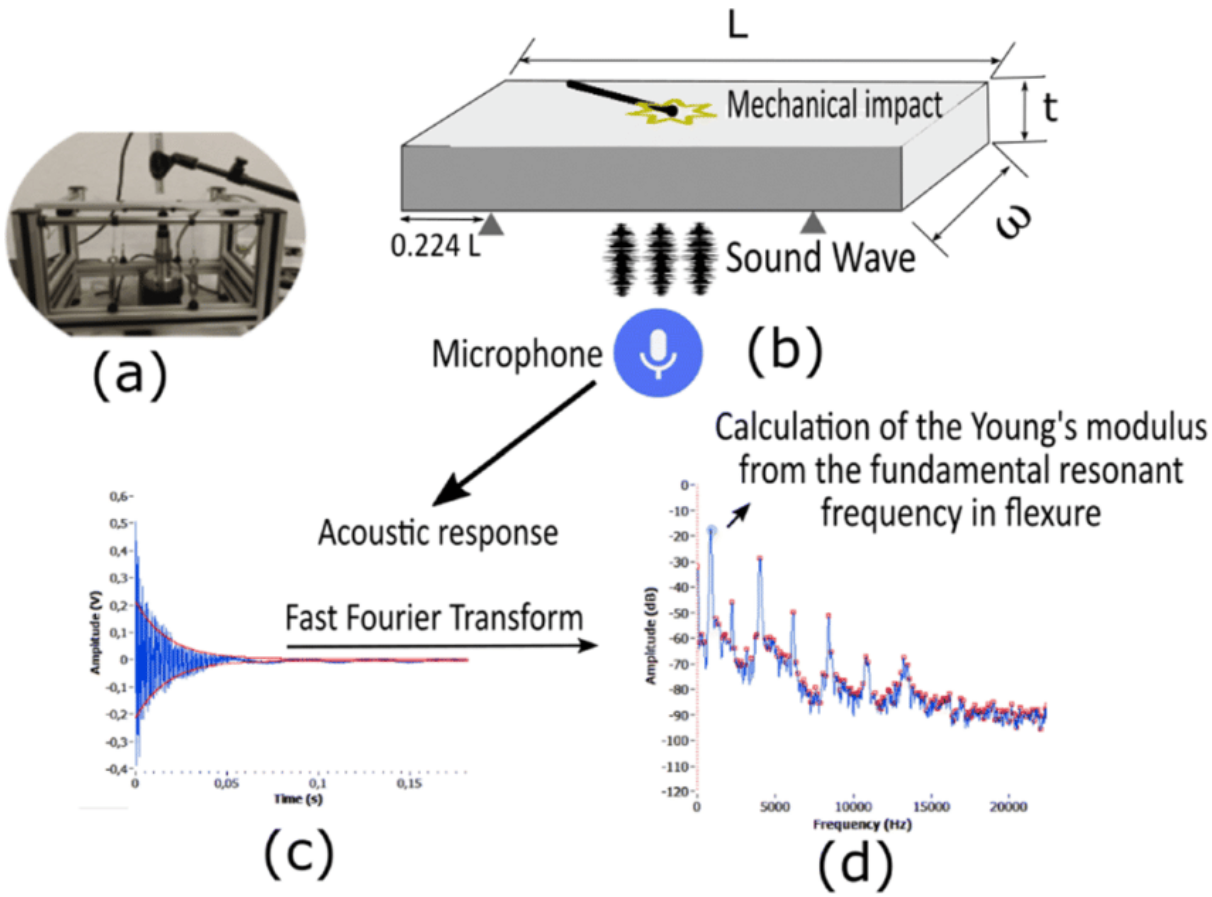


Figure 7 – Example set-up of the IET with resonance spectrum output [43]

2.3 Signal Processing Methods

This section investigates the signal processing techniques required and the effect a damaged component has on the response, which will help achieve objective 2a. The raw data measured from the sensors contains noise, which needs to be filtered before any processing of the data can be performed. For both sensors, a low pass or high pass band filter could be designed to remove as much noise as possible from the actual signal. This could be used to filter out any signal that is outside the frequency range of interest. MATLAB has a built-in application called Filter Designer, which can be used quickly to design appropriate filters for the raw signal.

The RUS method directly outputs the response in the frequency domain from two transducers and therefore the responses could be compared to identify the natural frequencies.

Conversely, the IET outputs the response in the time series domain. This signal must be passed through a Fast Fourier Transform (FFT) to identify the frequency components of the data. A FFT computes the Discrete Fourier Transform (DFT) of the time series signal and converts it into the frequency domain. This was highlighted by Fariñas, et al. [44], who successfully applied a FFT to the raw signal data from an air-coupled transducer to remove noise and calculate the natural frequencies in MATLAB.

Figure 8 displays an example of the raw signal data from a microphone and Figure 9 demonstrates the amplitude output from an FFT analysis on the raw signal, where the natural frequency modes are clearly identified by the red triangles.

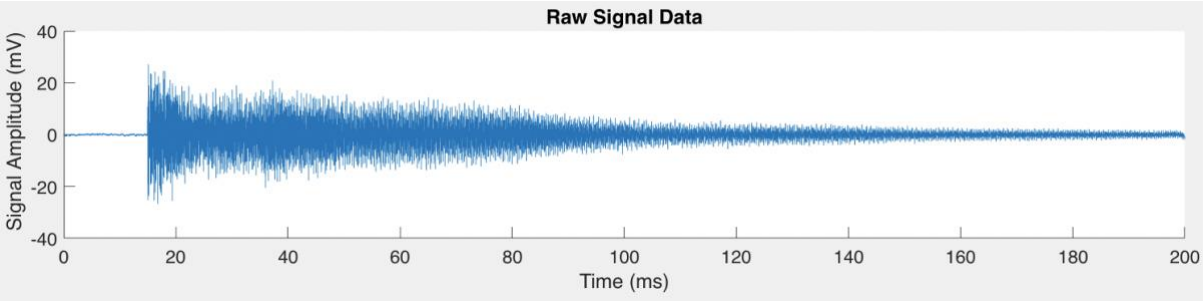


Figure 8 – Example of the raw microphone signal from an impact hammer test

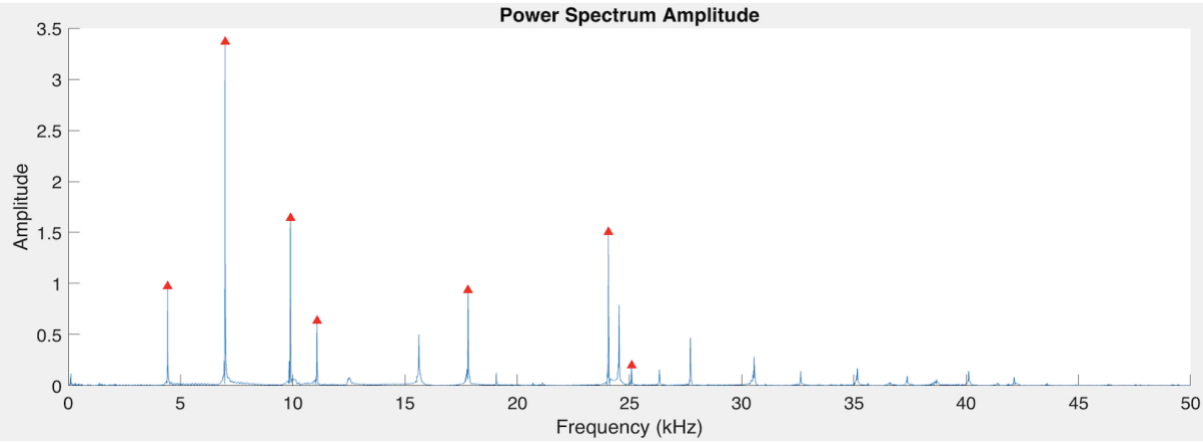


Figure 9 – Example of a power spectrum from an FFT of the raw microphone signal

Figure 10 clearly illustrates the impact of the increasing defect size on the natural frequency of a mode; shown as one of the red triangles in Figure 9. This direction change in frequency is the same for all the modes, as the whole response is shifted towards the left, due to the reduced stiffness of the component. The width of the peak can change as the defect size increases and therefore could also be used as an identifying feature.

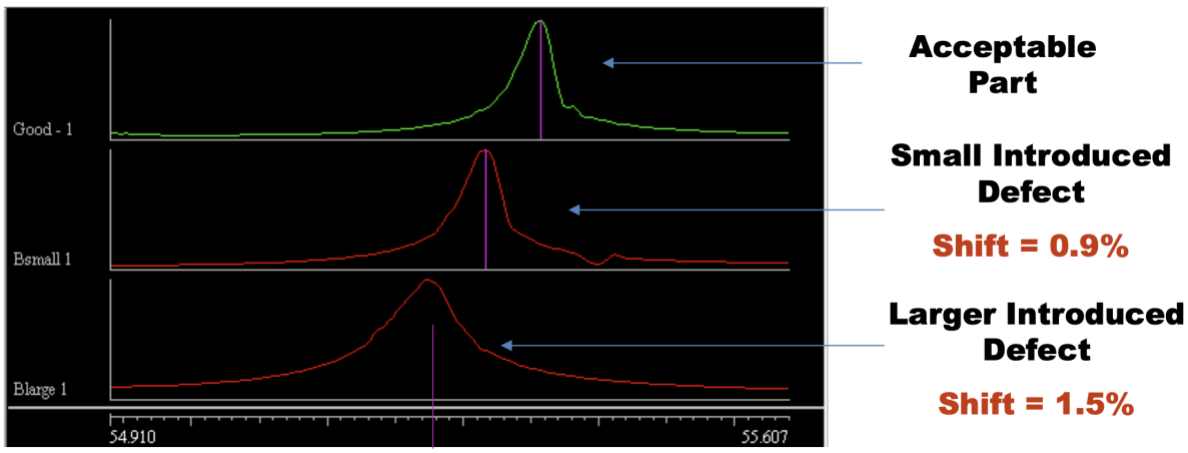


Figure 10 – Example change in natural frequency with increasing defect size [22]

2.4 Machine Learning Methods

This section investigates the different ML methods and reflects on which one would be most applicable for the project to achieve objectives 2b, 2c, 2f, 2g and 2h. ML is a subsection of AI that can automatically learn and improve through experience without being directly programmed. There is also a subset of ML called Deep Learning (DL) which can learn features from the data, instead of the features being specified by a human, and therefore the algorithm can self-train, see Figure 11.

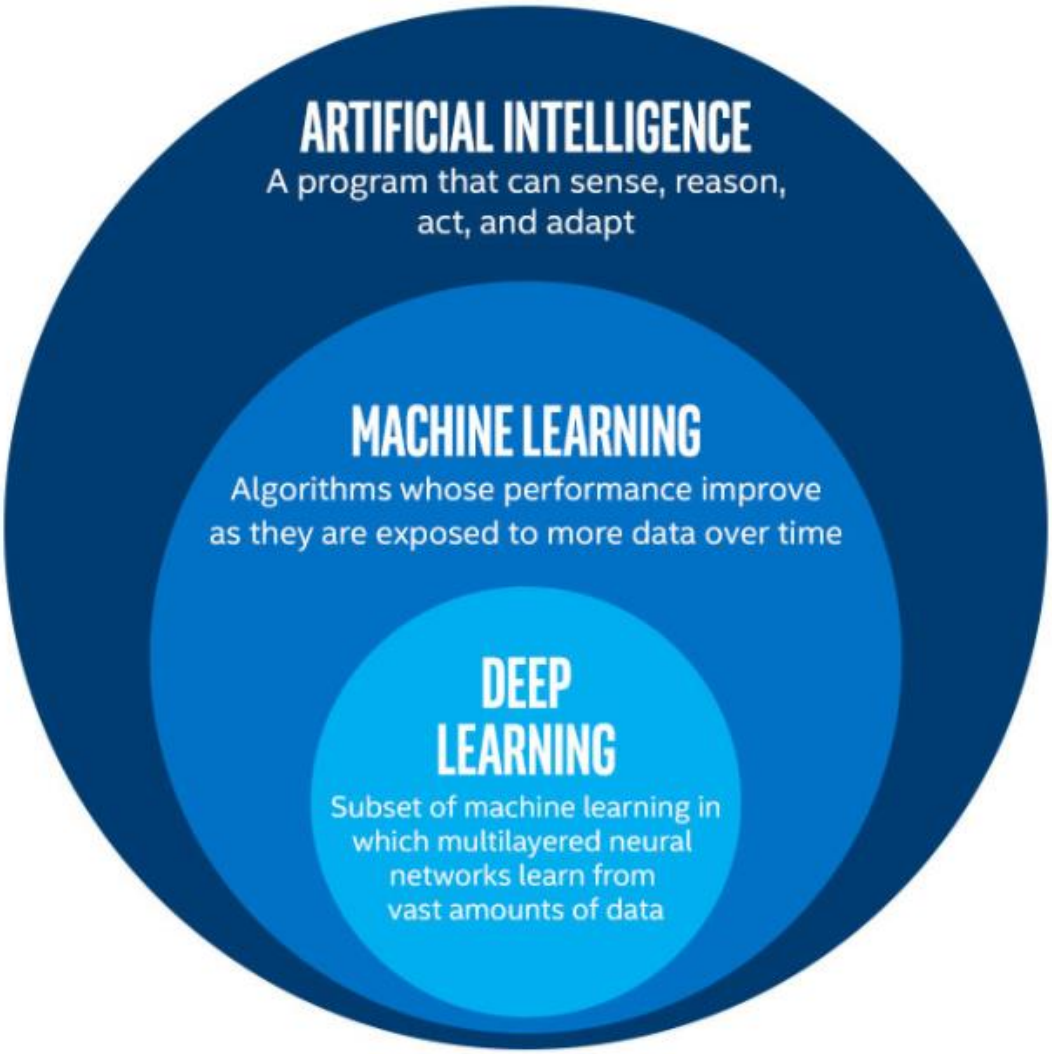


Figure 11 – Diagram of the breakdown of AI encompassing ML and DL [45]

There are four main ML methods which are explained with their common algorithms in Table 3. Two of these methods are relevant to the project and are explored further in Section 4. The blades are labelled damaged or undamaged, as their condition is known, allowing a supervised learning method to be applied. Within supervised learning, there are two subsets; namely classification and regression problems. Fumo [46] concluded that a classification problem outputs discrete or categorical values, such as a damaged or undamaged component, whilst a regression problem outputs a continuous variable. To achieve the aim of the project, a classification problem will therefore be used.

Table 3 – Different types of learning methods for ML [46], [47], [48], [49]

Learning Methods	Explanation	Common Algorithms
Supervised	• Creates a model using data that has both the inputs and outputs • New inputs are mapped to an output using the training data (predictive) • Types are classification or regression	• Discriminant analysis • Naïve Bayes • Decision trees • Linear regression • Support vector machines • k-nearest neighbours • Neural networks • Deep Learning
Unsupervised	• Creates a model using data that only has inputs; output is unlabelled • It looks for patterns in the data which are previously unknown (descriptive)	• k-means clustering • Spectral clustering • Gaussian mixture model • Density-based spatial clustering of applications with noise • Neural networks
Semi supervised	• A mix of supervised and unsupervised where some data has outputs and others do not	• A mix of supervised and unsupervised algorithms
Reinforcement	• Creates a model to state how agents should take action in an environment in order to minimise risk or maximise reward	• Q-Learning • Temporal difference • Deep adversarial networks

Until the exact details of the problem are known and what the main priorities are, such as training time, accuracy, or memory usage, the selection of algorithm cannot be made. For instance, support vector machines are quick to train, but require linear decision boundaries; whereas neural networks have a slower training time, but are accurate. The training time for DL is extremely slow but allows the user to simply input the raw signal and the known classification of the blade. Therefore, multiple algorithms will be used in the project to identify which one is most appropriate for the problem.

The standard workflow of a ML problem is seen in Figure 12.

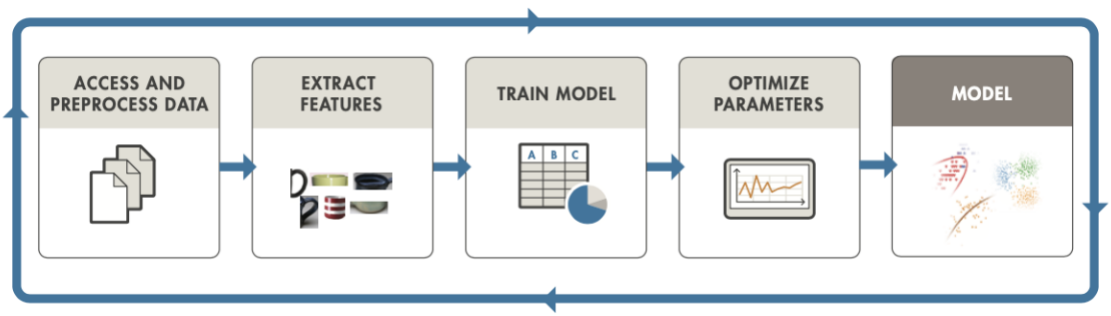


Figure 12 – The standard workflow for creating a ML model [48]

Machine learning has been successfully used with NDT methods by many researchers. Ghosh, et al. [50] used an artificial neural network with RUS when testing order parameter in URu_2Si_2. Fariñas, et al. [44] applied a convolutional neural network with non-contact RUS when testing relative water content in leaves. Paral, et al. [51] applied DL with IET when assessing steel frames. Finally, Charalampous, et al. [26] used a support vector machine to find fault diagnosis of an AM process, to cite just a few.

There are many programming languages which support ML. Python is the most popular due to the numerous supported libraries, such as TensorFlow, Keras, Theano, PyTorch and Scikit-learn, as highlighted by Innat [52] and D. Costa [53]. This view was supported by Nnamdi [54]. However, Miskuf, et al. [55] emphasized that the MATLAB Statistics and Machine Learning Toolbox can also be used as effectively in conjunction with a GUI.

2.5 Literature Review Conclusion

In conclusion, as a result of the literature review, although there are many NDT methods, only the RUS method and IET satisfy the aim of the project. It was noted that some methods only inspect the sub-surface and surface, whilst others highlight the presence of defects, but do not quantify the impact on the mechanical properties of the component. The IET has been selected for this project, compared to the RUS method, due to its simplicity to implement, only requiring a microphone and a simple impact hammer mechanism. The signal data will be passed through the MATLAB Signal Processing Toolbox to filter the signal and perform a FFT to identify the natural frequencies of the blade.

For the ML aspect of this project, a supervised classification algorithm has been primarily selected and will be implemented using the MATLAB Statistics and Machine Learning Toolbox. The rationale for this decision was simply because the test data obtained will be labelled, as it is known what the condition of the training blades are. The possibility of using an unsupervised algorithm with only undamaged blade data will also be investigated.

In summary, the project will consist of the creation of a test rig, using the non-destructive IET, with supervised and unsupervised ML algorithms to discern the classification of turbine blades quickly and reliably.

3 Test Rig Development

As a test rig was required to collect vibration test data for the ML algorithm, this section discusses the methodology applied, the key results and discussion of the main outcomes of the test rig development. Covid-19 restrictions were in place during the design of the rig. Consequently, it was decided to make the test rig as simple as possible such that it could be assembled using off-the-shelf components.

3.1 Mechanical Development Methodology

The test rig was developed using the systematic engineering design process to ensure that the new rig was an improvement on the previous rig used in preceding projects. The common stages of the process are; research, create design requirements, concept generation, detailed design, prototype testing and final production. At the concept generation stage, the optimal concept was selected using the Pugh method. Figure 13 presents a flow chart of the methodology followed. The results and analysis of the design are explained at each stage.

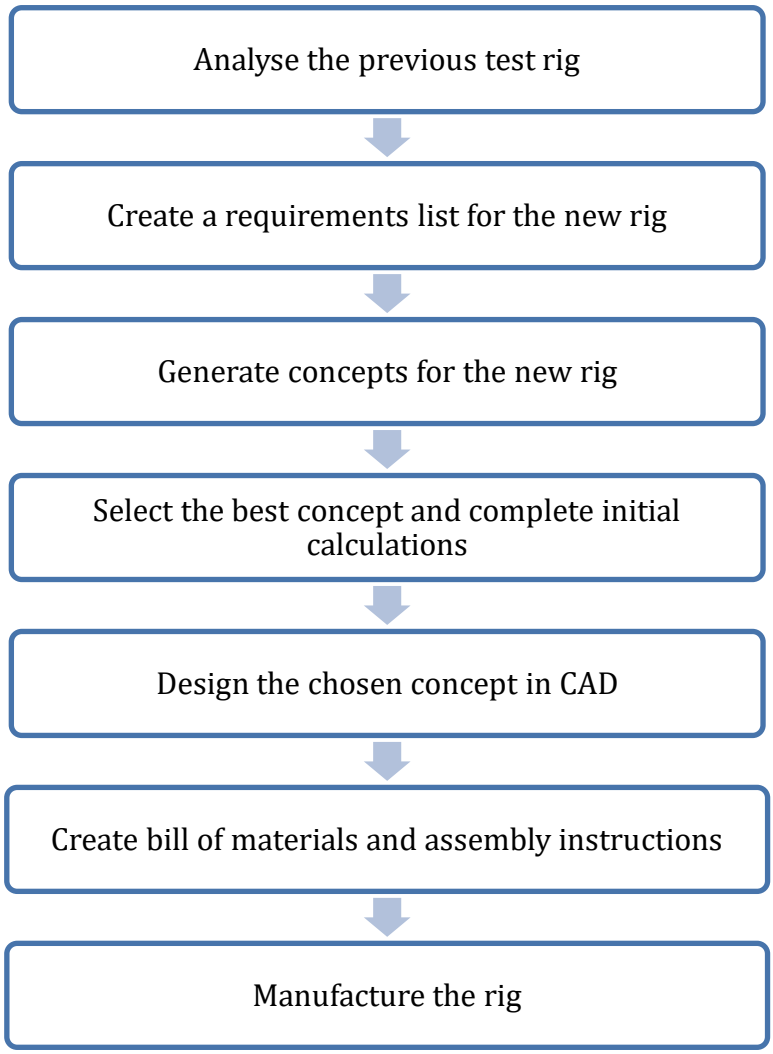


Figure 13 – Flow chart of the mechanical design methodology

3.2 Analysis of the Previous Rig

The previous rig was designed and built as a proof of concept rig and is presented in Figures 14a, 14b and 14c. The rig was fundamentally made up of two main areas; an enclosure which contained the blade and impact hammer, and the impact hammer mechanism.

Figure 14a shows the front view of the enclosure with the side panel in place and Figure 14b shows the inside of the enclosure and the location of the blade. The outer frame of the enclosure was made from Medium-Density Fibreboard (MDF), which had been glued together. The blade itself simply sat on the base piece of MDF.

The blade was originally chosen to not be rigidly fixed to the base piece of MDF, as a fixed blade would introduce extra natural frequency modes into the response. These natural frequency modes change depending on the stiffness of the mount, as seen by Equation 2, Section 2.1. The stiffness changes depending on the tightness of the screws in the mount leading to unrepeatable measurements, which is undesirable.

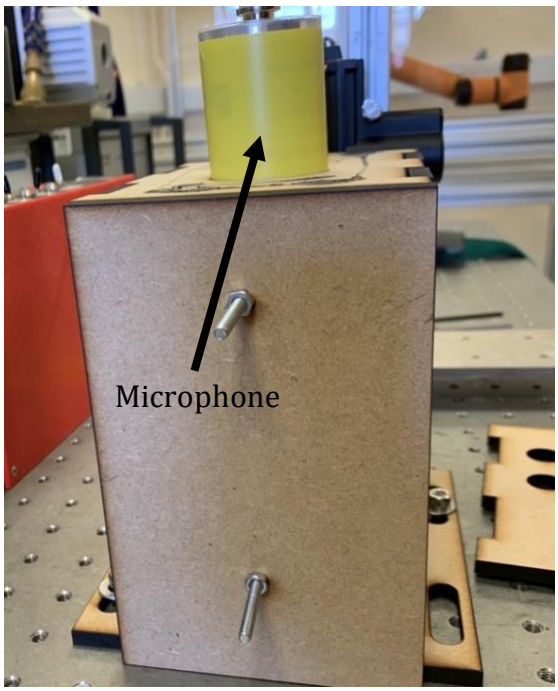


Figure 14a – Front view of the previous test rig

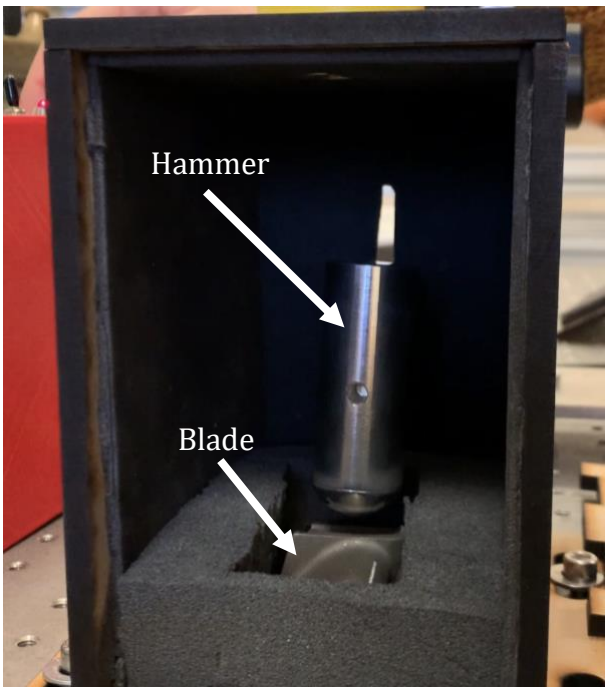


Figure 14b – Front view of the previous test rig with outer MDF removed

Figure 14c shows the side view of the previous rig showing the mechanism of the impact hammer. The hammer shaft is moved vertically by the linear actuator, which in turn moves the hammer vertically inside the enclosure. The end of the rod is fixed to the horizontal aluminium extrusions, outside the image, and acts as a hinge. The actuator, as well as gravity, create the impact force required onto the blade. In the previous rig, the force created by the actuator was slightly too high and resulted in minor damage to the blade after repeated impacts. The damage to the blades was unacceptable, as the rig and testing method needed to be non-destructive.

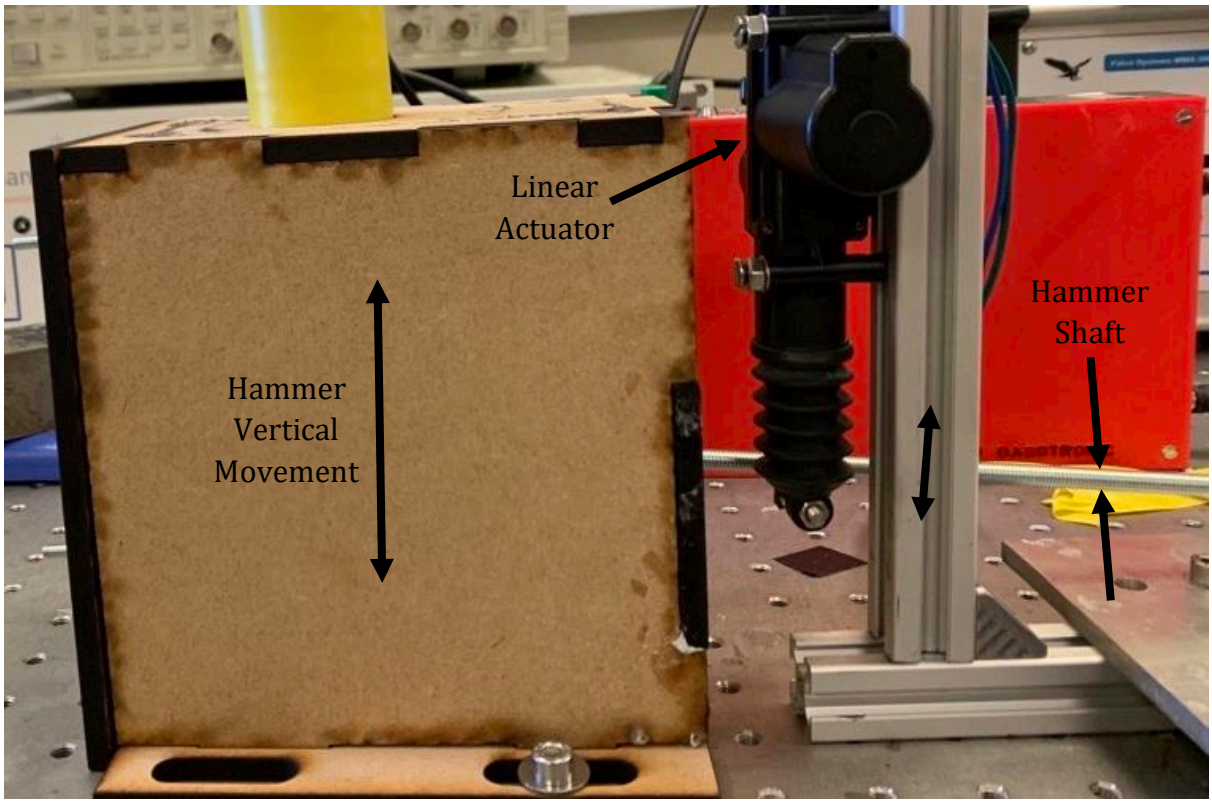


Figure 14c – Side view of the previous test rig showing the impact mechanism

Table 4 identifies the positives and negatives of the previous rig and was used as a starting point when the requirements list was produced. This was to ensure that the new rig had all the positives of the previous rig, whilst eliminating the negatives.

Table 4 – List of positives and negatives of the previous test rig

Type	Aspects
Positives	• Simple construction • Off-the-shelf components • Variable hammer impact height with movable actuator • Variable hammer tip • Achieves a single impact event
Negatives	• Not a rigid design as solely made from glued MDF • Actuator mechanism and enclosure are not connected, producing unsatisfactory repeatability • Minimum acoustic foam inside the enclosure, adding noise to the measurement • Blade bounces upon impact, thus reducing repeatability • Impact force is too high, leading to damage to the blades • Test must be conducted manually • Rig takes a long time to set up correctly

3.3 Requirements

Table 5 presents the mechanical requirements list for the rig. The importance of each requirement has been distinguished using the wish/must ranking and subsequent weighting system. The requirements that were musts with the greatest weight (where 3 is the most important) were prioritised. In addition, each requirement had a success criteria which was quantifiable where appropriate.

Table 5 – Mechanical requirements list

	Requirements	Must/ Wish	Wt. 1-3	Success criteria	Date	Reference
1	Under budget	Must	3	£250 maximum	01-Feb	Roger Ngwompo
2	Hammer strikes the same location	Must	3	Hammer is horizontally constrained	09-Feb	Marco Boccaccio
3	Repeatable impact event	Must	3	Hammer is dropped from the same height	09-Feb	Marco Boccaccio
4	Rig does not interfere with impact event	Must	3	20+ mm thickness of foam on the inside	09-Feb	Marco Boccaccio
5	Hammer only strikes once	Must	3	Mechanism to prevent a double impact	08-Feb	Michele Meo
6	Larger blades can be tested	Must	3	Internal footprint is at least 100 x 100 mm	22-Feb	Nick Thomas
7	Correct focal length between sensor and blade	Must	3	Microphone is at least 76 mm away from the blade	22-Feb	Marco Boccaccio
8	Blade is fixed in place	Must	2	Base of blade is surrounded by foam	08-Feb	Michele Meo
9	Rig is safe	Must	2	Rig safety in line with British Standards	10-Feb	Nick Thomas
10	Hammer is removable	Must	2	The rig can be easily accessed	10-Feb	Marco Boccaccio
11	Test data is automatically processed	Must	2	Entire rig can interface with MATLAB	09-Feb	Marco Boccaccio
12	Simple design	Wish	1	Use standard off-the-shelf parts	10-Feb	Nick Thomas
13	Quick to manufacture and assemble	Wish	1	Rig is operational within a few weeks	10-Feb	Nick Thomas
14	Mounting points for multiple microphones	Wish	1	Top MDF piece can be switched	09-Feb	Marco Boccaccio
15	Rig is isolated from the surroundings	Wish	1	External noise and vibration are reduced	10-Feb	Nick Thomas
16	Tests are conducted quickly	Wish	1	Full test can be completed in a few minutes	10-Feb	Nick Thomas
17	Tests have little human intervention	Wish	1	Test is autonomous after it has been setup	10-Feb	Nick Thomas
18	Accommodates different blade sizes	Wish	1	Foam can be easily removed and replaced	10-Feb	Nick Thomas
19	Fits easily on a workbench	Wish	1	0.15 m^2 rectangle or less	10-Feb	Nick Thomas

The requirements list comprised of key factors which had to be achieved as primary objectives. These were, for example, an enclosure filled with acoustic insulation on the interior; an impact hammer mechanism, which would be connected to the enclosure, have a tuneable impact force, and prevent double impacts. Finally, the test rig had to be automated.

3.4 Concept Generation

Concepts were generated for the new rig, which were derived from the key factors on the requirements list. The two main functions that the rig needed to perform were to produce a repeatable single impact with the correct force and to absorb reflections from the impact. These functions are explored below with the relevant concepts and accompanying explanations.

3.4.1 Enclosure Design

The previous rig had acoustic foam only on the bottom of the enclosure and was primarily used to hold the blade in place. The remaining internal walls were made of MDF layered with thin foam, which combined to form a relatively dense material with a small sound absorption coefficient [56]. This resulted in a reasonable amount of sound being reflected into the enclosure, thus adding secondary waves and noise to the microphone, which is undesirable. Furthermore, the previous design was not very rigid, as the frame was made from glued MDF, and the hammer mechanism was not directly connected to the enclosure.

To improve the sound absorption, thick acoustic foam was adhered to all the inside faces of the MDF, as it has a much higher sound absorption coefficient, approximately 0.8, and therefore absorbs a greater amount of sound, which minimises reflections [57]. To improve the rigidity and repeatability of impacts, a frame made from aluminium extrusions was designed to house the pieces of MDF, the hammer mechanism and microphone.

3.4.2 Impact Hammer Mechanism

The most important factor with any impact hammer test is the impact event itself. This is because it is the excitation which produces the vibration in the blade and consequently sound, which is later used with a FFT to identify the natural frequencies. The impact must be clean and must have enough force to adequately excite the blade, whilst only allowing a single impact on the blade, as double impacts are detrimental to the response. The impact mechanism must also be controlled by software to allow the test rig to run autonomously.

Nine mechanism concepts for the impact hammer mechanism were created and are presented below.

3.4.2.1 Concept 1

The first concept used a stepper motor attached to a pivot which moved the hammer to an angle and therefore the height required to impact a vertical blade, see Figure 15. To prevent a double impact, the power to the stepper motor could be turned on after the impact, to hold the hammer in place.

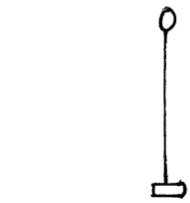
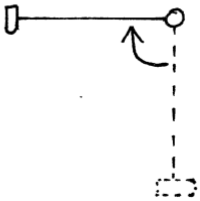
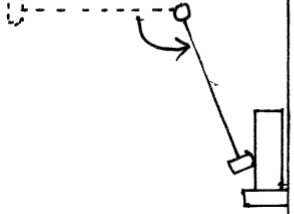
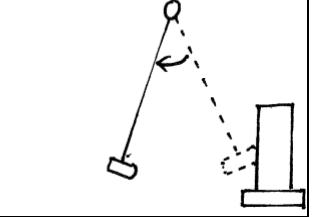
1	2	3	4
			
Gravity is used to find the datum point	The stepper motor moves the required angle anticlockwise	The power is cut from the motor and the hammer drops due to gravity	After impact the power is restored to hold the hammer in place

Figure 15 – Mechanism sketch and explanation of concept 1

3.4.2.2 Concept 2

The second concept used a DC motor to turn a semi toothed gear meshed with a fully toothed gear which was attached to a pivot, to move the hammer to the required height, see Figure 16. An alternative to the two gears would be the use of a cam and follower mechanism.

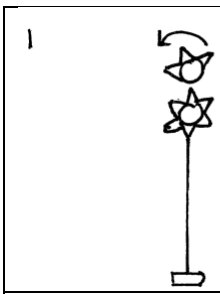
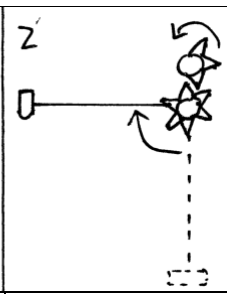
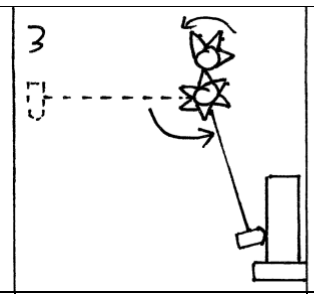
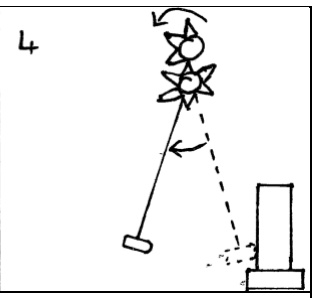
1	2	3	4
			
Gravity is used to find the datum point	A DC motor drives the partial gear which meshes with the full gear and then rotates the hammer	The motor continues to move until the partial gear no longer contacts with the full gear and then the hammer drops	The partial gear meshes with the full gear again, thus stopping the hammer

Figure 16 – Mechanism sketch and explanation of concept 2

3.4.2.3 Concept 3

The third concept used a DC motor to function as a winch to pull the hammer up towards the required height, see Figure 17. The DC motor would be positioned so the hammer would achieve the required height. To prevent a double impact, the power to the motor would be turned on again briefly to retract the string by a small amount, thus restricting the swing back motion of the hammer.

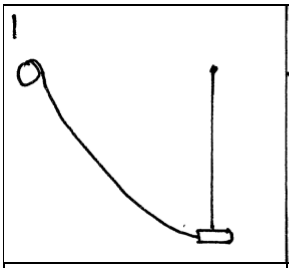
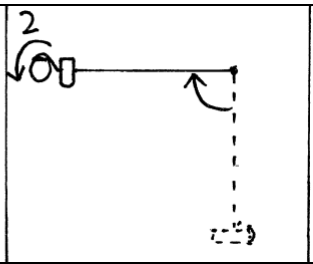
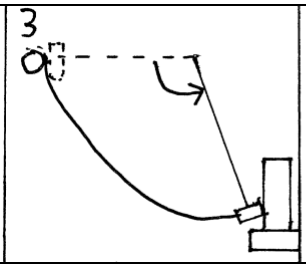
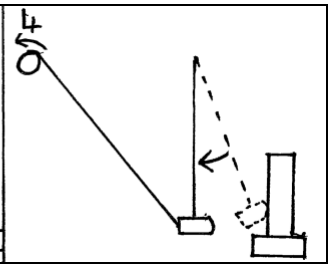
			
Gravity is used to find the datum point	The DC motor rotates to pull the string in by a set amount to rotate the hammer	The power is cut from the motor and the hammer drops due to gravity	The motor is turned on after impact to reduce the length of the string

Figure 17 – Mechanism sketch and explanation of concept 3

3.4.2.4 Concept 4

The fourth concept used a linear actuator with a hinged arm to pull the hammer at the end of the hinge up and down, see Figure 18. The actuator would be moved along the hinged arm to change the starting height and therefore impact force. The actuator would be configured to either pull down the arm with the help of gravity or to be switched off, resulting in gravity alone creating the impact force.

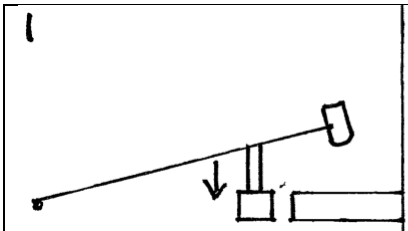
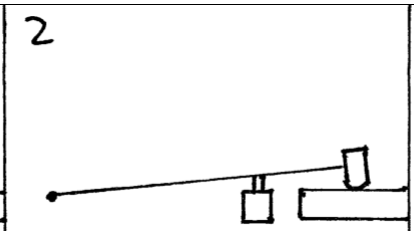
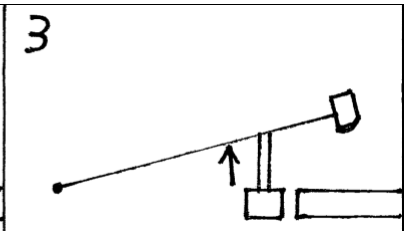
		
The linear actuator retracts, or the power is cut to allow the hammer to fall under gravity	The impact with the blade takes place	The actuator then extends pushing the hammer upwards

Figure 18 – Mechanism sketch and explanation of concept 4

3.4.2.5 Concept 5

The fifth concept used a linear actuator with a lever mechanism to pivot the hammer up and down at the end of the arm, see Figure 19. Similarly to concept four, the actuator could be moved along the arm to increase the starting height of the hammer, as well as using the actuator as the main method to generate the force.

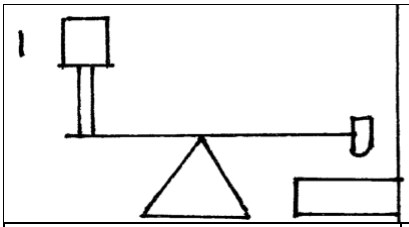
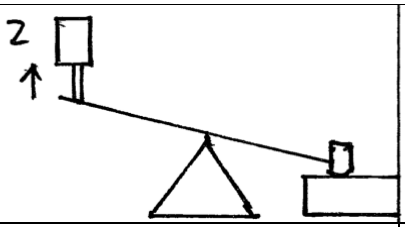
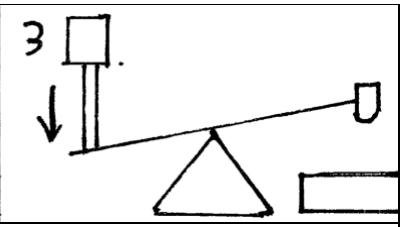
		
The linear actuator is stationary	The actuator retracts or the power is cut, allowing the hammer to fall under gravity	The actuator extends, thus lifting the hammer upwards

Figure 19 – Mechanism sketch and explanation of concept 5

3.4.2.6 Concept 6

The sixth concept used a linear actuator which was mounted above the blade, with the hammer tip connected to the end to create the impact, see Figure 20. The position of the actuator would need to be adjusted for each different blade to ensure that the end of the actuator stroke was only just impacting the blade. The impact force would be determined directly from the specification of the actuator.

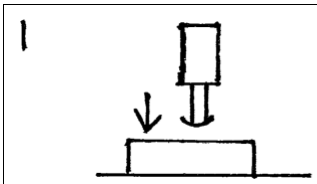
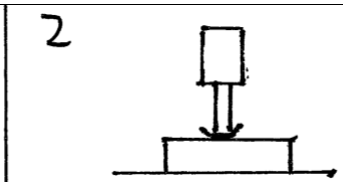
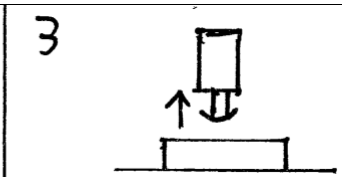
		
The linear actuator extends to impact the blade	The impact with the blade takes place	The actuator retracts pulling the hammer upwards

Figure 20 – Mechanism sketch and explanation of concept 6

3.4.2.7 Concepts 7, 8 and 9

Concepts 7, 8 and 9 are identical to concepts 4, 5 and 6 respectively, but the impact would occur at the bottom of the blade instead of at the top. If this were the case, then the blade would need some support on its top surface to prevent it from being pushed upwards.

3.5 Concept Convergence and Selection

To select the optimum impact mechanism, the Pugh method was used with the criteria from the mechanical requirements list and main features from Section 3.4. Furthermore, Table 6 was created to summarise the advantages and disadvantages of each impact hammer mechanism concept.

Table 6 – List of advantages and disadvantages for each impact hammer mechanism

Concept	Advantages	Disadvantages
1	• Stepper motor can easily rotate to a set angle • Easy to prevent double impacts	• Blade must be adequately supported upright • Repeated impacts may damage the motor
2	• Motor is not damaged by impacts	• Blade must be adequately supported upright • Partial gear must be changed for different heights • No clear method to prevent double impacts
3	• Easy to prevent double impacts	• Blade must be adequately supported upright • Height is fixed by the location of the motor • Hard to determine the exact location of the arm
4	• Easy to change the drop height • Very simple mechanism	• May need a spring to prevent a double impact
5	• Easy to change the drop height • Very simple mechanism	• May need a spring to prevent a double impact
6	• Easy to prevent double impacts	• Actuator must be carefully selected so it provides the optimum impact force
7	• Easy to change the drop height	• May need a spring to prevent a double impact • Something is required to constrain the blade vertically
8	• Easy to change the drop height	• May need a spring to prevent a double impact • Something is required to constrain the blade vertically
9	• Easy to prevent double impacts	• Actuator must be carefully selected so it provides the optimum impact force • Something is required to constrain the blade vertically

The Pugh method uses a concept as a datum, which acts as the baseline concept, and is scored 0. The other concepts were scored either -1, 0 or 1 signifying whether they were better or worse than the datum concept for the requirement criteria. The datum concept selected was concept 4 because it was similar to the mechanism used on the previous rig. The results of the other concepts can be seen in Table 7.

Table 7 – Results from the Pugh method for selecting the best concepts

Criteria/ Concept	1	2	3	4	5	6	7	8	9
Prevent double impacts	1	-1	1	DATUM	0	1	1	1	1
Variable impact force	0	-1	-1		0	-1	0	0	-1
Clean impact	-1	-1	-1		0	0	0	0	0
Easy to support the blade	-1	-1	-1		0	0	-1	-1	-1
Off-the-shelf components	0	-1	-1		0	0	0	0	0
Simple design	0	-1	0		0	0	-1	-1	-1
SCORE	-1	-6	-3	0	0	0	-1	-1	-2

From the analysis in Table 7, the best concepts were concepts 4, 5 and 6 as they had the highest scores. However, concept 6 was not flexible because the selected actuator would only be able to produce a single force and would need to be moved depending on the thickness of the blade.

Concepts 4 and 5 had the potential of being able to change the impact force. They also had a very simple design using off-the-shelf components to impact a horizontally mounted blade from the top, resulting in a simple constraining mechanism.

3.6 Initial Calculations and Component Selection

Before the detailed design stage, initial calculations were conducted, and components specified for the new rig. The main calculations required were to determine the impact force needed to adequately excite the blade, to find the best mechanism to create the force and finally to calculate the thickness of foam required to absorb the range of frequencies that were being measured.

3.6.1 Impact Force Required

The magnitude of the impact force directly contributes to the response the microphones measure and therefore the greater the impact, the larger the response is. A larger response is beneficial because the signal to noise ratio is greatly increased, allowing the frequency peaks of interest to be more easily identified and extracted. Furthermore, the microphones do not need to be as sensitive or have as high a resolution, which generally corresponds to a lower cost. Conversely, an impact force which is too large damages the blade and renders the test destructive instead of the intended non-destructive test. Therefore, an impact hammer test was conducted to ascertain the optimum force required which would produce a large response without permanently damaging the blade.

Figure 21 shows the experimental set-up of the test to identify the adequate force required to excite the blade and Figure 22 highlights the results of the test. The hammer tip has an accelerometer built in, which is used with Newton's second law to calculate the force of the impact, the blue line in Figure 22. The orange line is the response the microphone measures, which can be passed through an FFT to identify the natural frequency modes of the blade. The output from the FFT can be seen in Figure 23, which clearly highlights the natural frequency modes, demonstrating that the impact had the required force. This resulted in the ideal impact force of approximately 125 N.

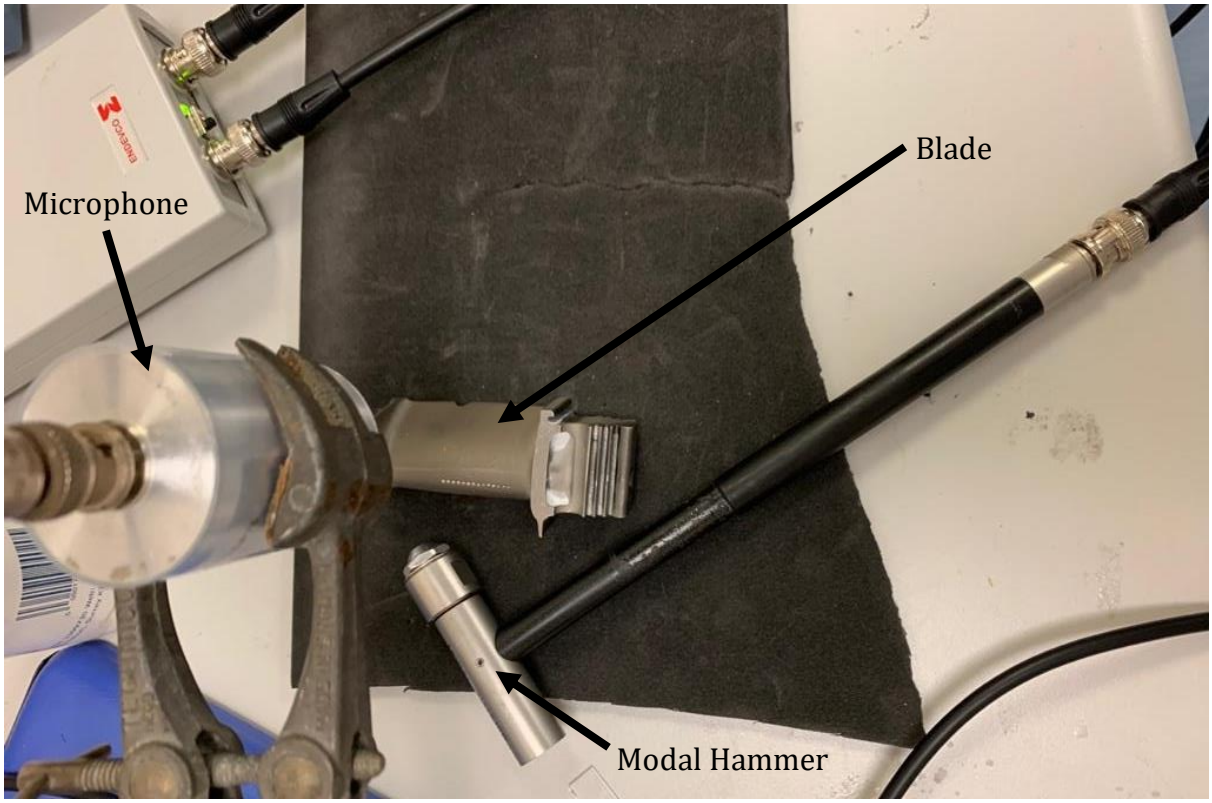


Figure 21 – Experimental set-up of the modal impact hammer test

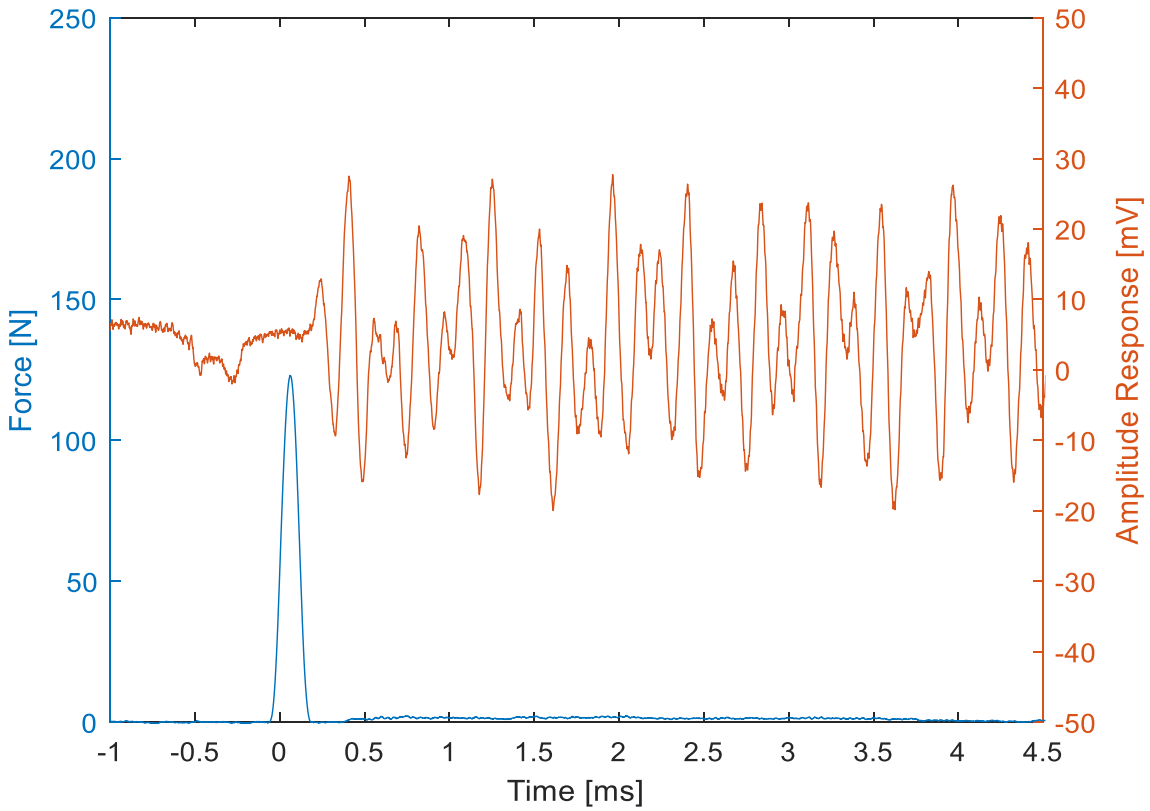


Figure 22 – Results from the impact hammer accelerometer and microphone

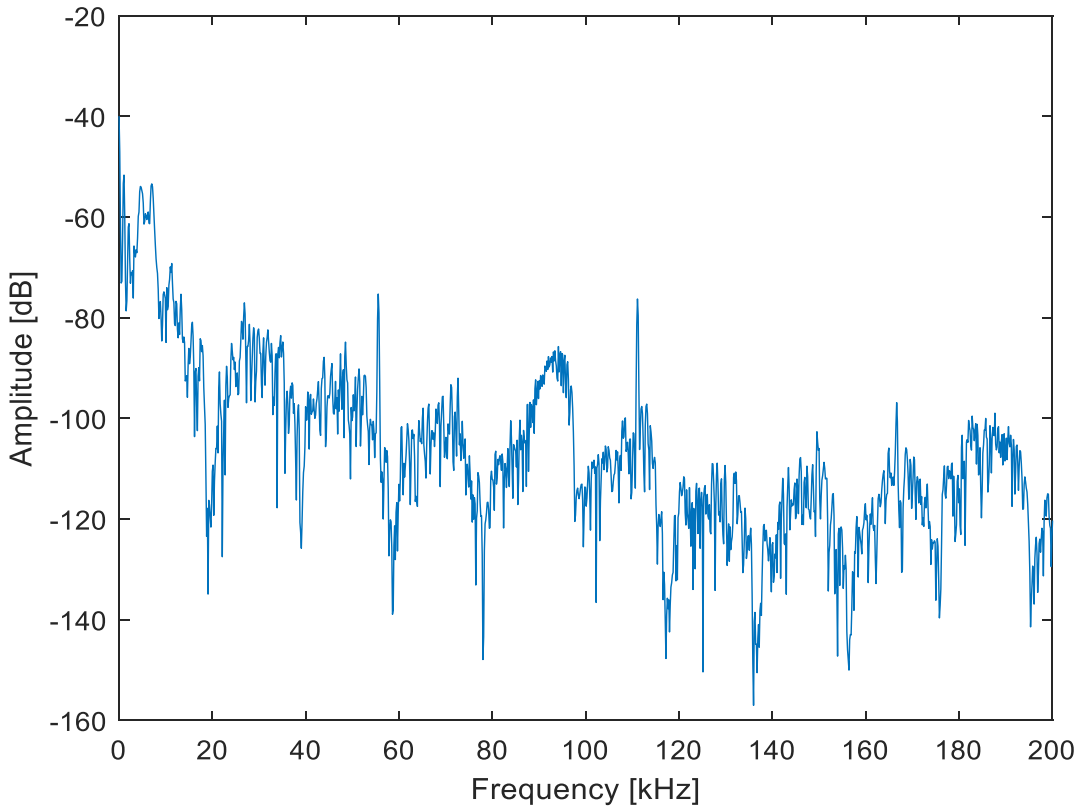


Figure 23 – FFT amplitude plot of the raw microphone signal

The impact force must either come from gravity alone or from the combination of the actuator and gravity. If the force solely comes from gravity, then the mass of the hammer tip and drop height must be determined. As described below, due to limited knowledge of the blades and the system, the mass and height were complex values to calculate and therefore the rig was calibrated after it had been constructed.

3.6.1.1 Hand Calculations

The conservation of energy states that the total energy of a system remains constant. This means that all the gravitational potential energy the hammer has at the drop height must be converted into elastic strain energy and work done energy. This is shown in Equation 3 and is expanded further in Equation 3.01.

$$GPE = U + WD \tag{3}$$

$$mgh = \frac{1}{2}V\sigma\varepsilon + Fd \tag{3.01}$$

The gravitational potential energy and work done could easily be calculated, however the strain energy could not be quantified, due to the complexity of the geometry of the blade. This meant that it was not possible for this calculation to be performed in this way and additional information from the CAD model of the blade would be required.

3.6.1.2 Simulink Contact Model

A simplified Simulink model was also created to investigate the magnitude of the impact force by changing the impact hammer mass and drop height, Figures 24 and 25. Unfortunately, the results of the model were unusable, as the model required the stiffness and damping values of the blade. These parameters were not possible to analytically calculate because the geometry of the blade was complex and experimentation to calculate these values were outside the scope of the project. Furthermore, the project owner's supervisor did not have a CAD model of the blade and therefore Finite Element Analysis (FEA) could not be used to estimate these values.

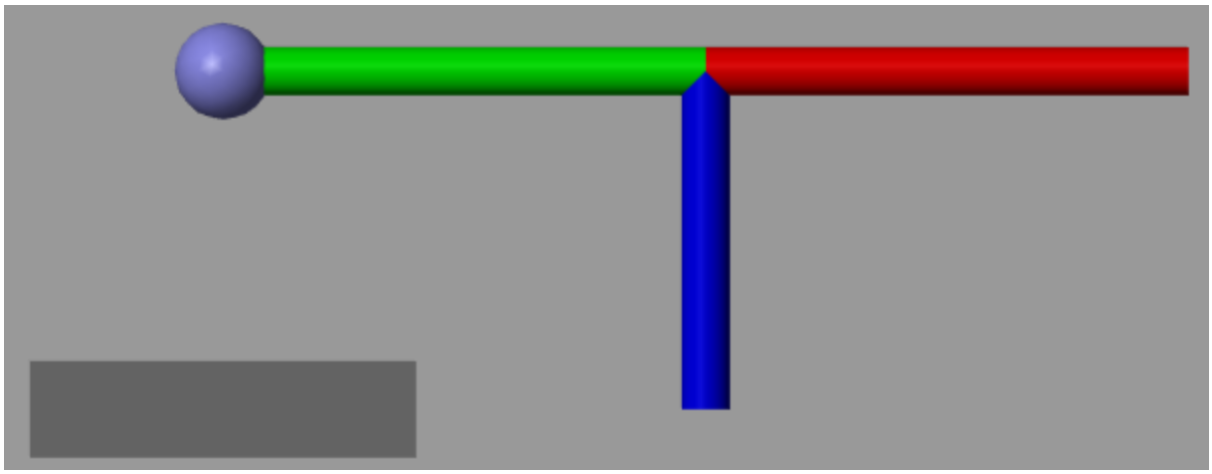


Figure 24 – Simplified Simulink model of the hammer impact at the drop height

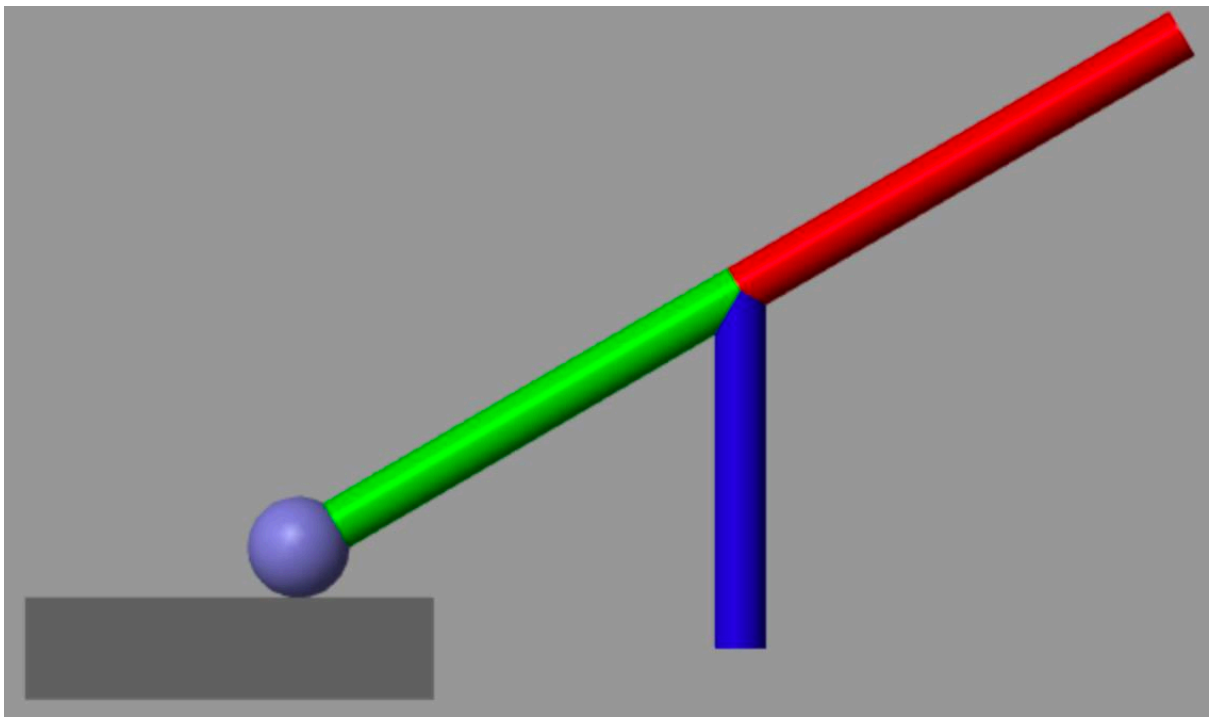


Figure 25 – Simplified Simulink model of the hammer impact after the impact

3.6.2 Foam Thickness Required

From the initial impact results and discussion, it was determined that the acoustic foam used should be able to effectively absorb sound waves with a frequency of 5 kHz and above, as this is the range of natural frequencies that were being measured. To be able to absorb sound waves effectively, the foam must have the required minimum thickness [58]. It must have the same or greater thickness than a quarter of the wavelength of the frequency it must absorb, using the simple quarter wavelength rule [59]. Equation 3.02 shows that the minimum thickness required is 17.2 mm using the wave equation [60]. Therefore, a standard 25 mm thick polyethylene acoustic insulation foam was selected, as it could be easily obtained off-the-shelf at low cost.

$$t = 0.25 * \lambda = 0.25 * \frac{v}{f} = 0.25 * \frac{343}{5{,}000} = 17.2\ mm \qquad (3.02)$$

3.7 Detail Design in Computer-Aided Design

The test rig was designed using the CAD package, Autodesk Inventor 2021. The model was designed using best practice methods to ensure that the model could be easily modified by another engineer; the parts and assembly can be found on GitHub.

3.7.1 Internal Test Volume

The specified microphone had a focal length of 76 mm, hence this was the minimum distance needed between the bottom of the microphone and to where the impact would occur. The footprint of the blade was 80 x 35 mm, as seen in Figure 26. To allow for the potential of being able to evaluate larger blades, the chosen width and length was 150 x 100 mm. This also ensured that the opening to the test rig was wide enough for an engineer's hand. Moreover, the base piece of foam was designed so that it could be slid out easily and replaced with a new piece, to house a different sized blade.

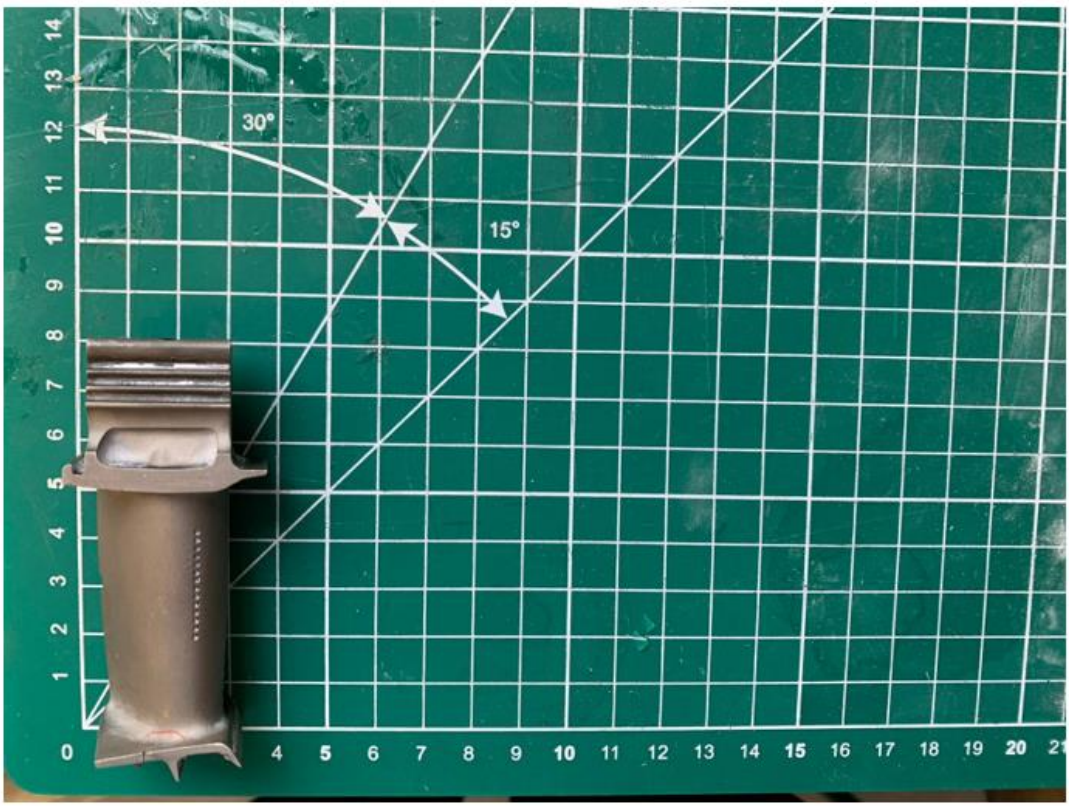


Figure 26 – Footprint dimensions of the turbine blade under test

3.7.2 Enclosure Design

The enclosure was then built from the internal test volume of 150 x 100 x 100 mm, using MDF as a base on which to adhere the foam. The MDF was sized to fit into the recess in the aluminium extrusion, producing a very rigid design. Furthermore, the recess allowed the front piece of MDF to slide up and down to function as an opening hatch, which allowed for quick and easy access, see Figure 27. The top piece of MDF was designed to be secured to the aluminium extrusion, ensuring that the microphone was always in the same position, thus increasing the repeatability of the measurements. Furthermore, if a different microphone or several small microphones were to be used instead, the top piece could be easily replaced.

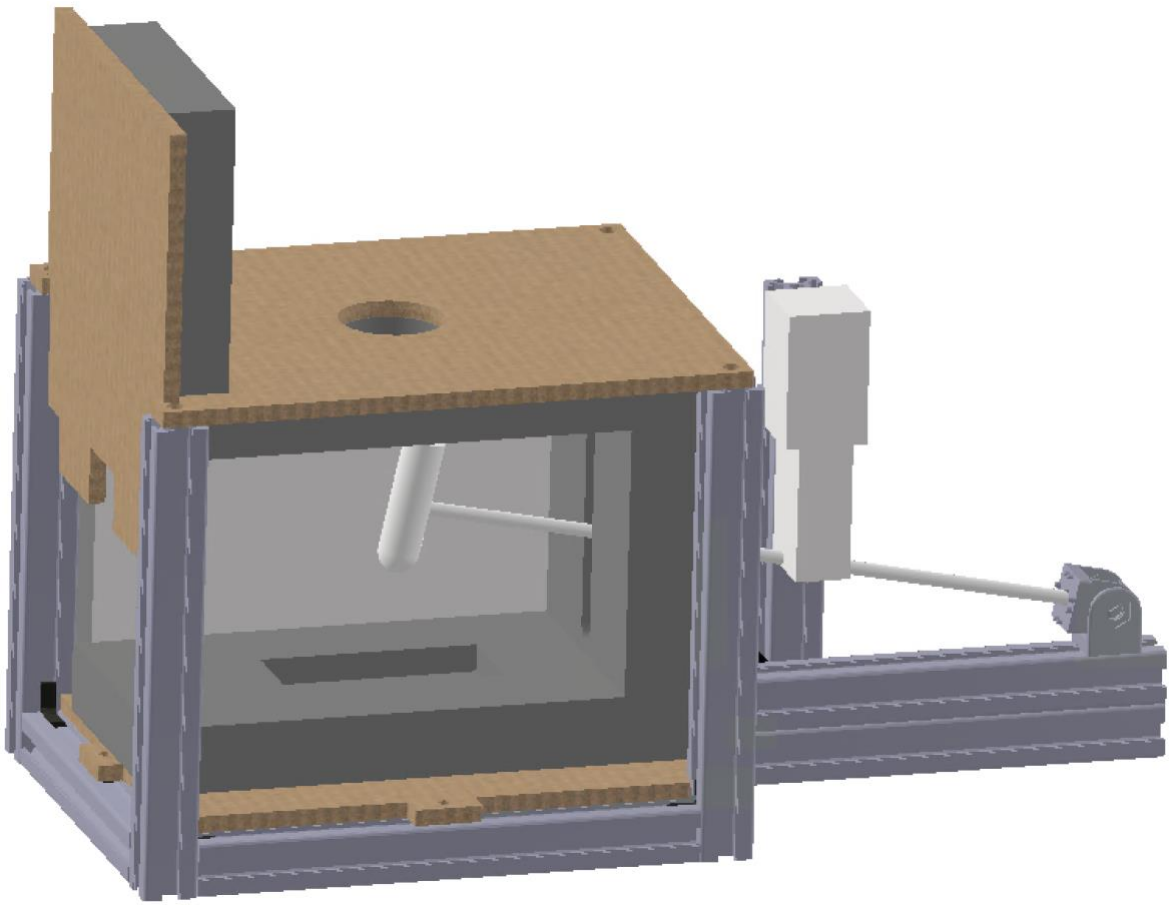


Figure 27 – Side view of the new test rig demonstrating the key features of the rig

3.7.3 Impact Hammer Mechanism

The two chosen concepts with the key requirements were transferred to the electronic technicians to specify the electronic components, create the circuitry, and control system, as the project owner was unable to do this due to the restrictions. The mechanism would be controlled by an Arudino which was connected to the MATLAB application.

3.8 Bill of Materials

Table 8 contains the bill of materials for the new rig, which totalled £157.91, comfortably under the £250 budget. This did not include the cost of the microphone and oscilloscope, as these were sourced from the previous rig, or general electronic components, such as the stepper motor.

Table 8 – Bill of materials for the new test rig

Item	Quantity	Supplier	Part Number	Unit Cost £	Total Cost £
Oscilliscope	1	Fanrell	PP479	949	949
Big mic	1	Ultrangroup	NCG100-D25-P76	-	-
Small mic	9	Farnell	2362676	3.22	28.98
Strut 3000mm	1	RS	3842992888/3000	22.1	22.1
Right Angle out	15	RS	3842523511	2.58	38.7
Right Angle in	10	RS	3842535574	1.176	11.76
T slot nut	10	RS	3842523135	0.595	5.95
Pivot joint	2	RS	180-9134	21.63	43.26
Foam 25mm	1	RS	103-4068	36.14	36.14

3.9 Solution Specification

As of this report being submitted, the rig had been constructed but no testing or calibration had been carried out. The design was compared to the requirement list to ascertain if the requirements had been met and this is summarised in Table 9. The results of the test rig itself and software application are discussed in Section 5.

Table 9 – Solution specifications compared to the mechanical requirements list

	Requirements	Success criteria	Success/ fail	Actual
1	Under budget	£250 maximum	£157.91	✓
2	Hammer strikes the same location	Hammer is horizontally constrained	A little	✗
3	Repeatable impact event	Hammer is dropped from the same height	Yes	✓
4	Rig does not interfere with impact event	20+ mm thickness of foam on the inside	25 mm	✓
5	Hammer only strikes once	Mechanism to prevent a double impact	Yes	✓
6	Larger blades can be tested	Internal footprint is at least 100 x 100 mm	100 x 150 mm	✓
7	Correct focal length between sensor and blade	Microphone is at least 76 mm away from the blade	100 mm	✓
8	Blade is fixed in place	Base of blade is surrounded by foam	Yes	✓
9	Rig is safe	Rig safety in line with British Standards	Yes	✓
10	Hammer is removable	The rig can be easily accessed	Yes	✓
11	Test data is automatically processed	Entire rig can interface with MATLAB	Yes	✓
12	Simple design	Use standard off-the-shelf parts	Yes	✓
13	Quick to manufacture and assemble	Rig is operational within a few weeks	11 weeks	✗
14	Mounting points for multiple microphones	Top MDF piece can be switched	Yes	✓
15	Rig is isolated from the surroundings	External noise and vibration are reduced	Yes	✓
16	Tests are conducted quickly	Full test can be completed in a few minutes	Yes	✓
17	Tests have little human intervention	Test is autonomous after it has been setup	Yes	✓
18	Accommodates different blade sizes	Foam can be easily removed and replaced	Yes	✓
19	Fits easily on a workbench	0.15 m^2 rectangle or less	0.09 m^2	✓

The new test rig met 17 out of the 19 requirements, concluding that it was a successful design. The rig was manufactured and assembled within two weeks after it had been fully designed, which was a testament to its efficient design. However, it took 11 weeks to reach that stage as the electronic technicians were not readily available. Furthermore, the hammer shaft was not completely constrained in the horizontal axis, as it was simply held by the acoustic foam in the cut-out channel of the MDF. This may lead to a slightly varying impact location, thus reducing the repeatability.

4 Software Development

The user required a GUI to create, train and use the ML algorithms to classify new blades. This section discusses the methodology applied, the key results and discussion of the main outcomes of the application development. The different types of ML and the approaches taken are explored, as well as explaining the architecture of the GUI and sub-functions. Due to the Covid-19 restrictions, a greater emphasis was placed on the GUI and ML aspect of the project.

4.1 Software Development Methodology

The software application to be used alongside the new test rig was developed using the software design process. This process was similar to the engineering design process where requirements were created, the general architecture was decided, and the modules and fundamental functions were written. Figure 28 shows a flow chart of the methodology used, with each step describing the results and analysis of the software at that stage.

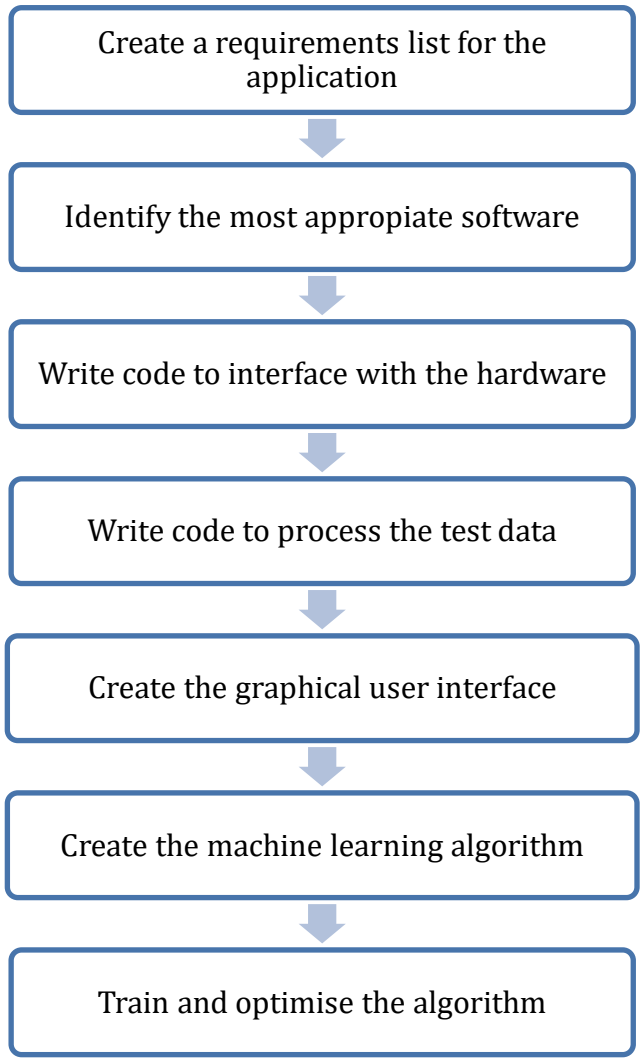


Figure 28 – Flow chart of the software design methodology

4.2 Requirements

Table 10 is the software requirements list for the rig. It was constructed using the same method as the mechanical requirements list. The key requirements of the application were: its ease of use, its autonomous nature, and its capabilities to not only classify a new blade, but also its ability to collect new training data quickly, as well as being able to retrain the ML algorithms. The GUI needed to be as simple as possible, with the complicated ML parameters being automatically optimised.

Table 10 – Software requirements list

	Requirements	Must/ Wish	Wt. 1-3	Success criteria	Date	Reference
1	Test is autonomous	Must	3	One button to run test	10-Feb	Nick Thomas
2	Control the test rig	Must	3	Code can interface with the hardware	09-Feb	Marco Boccaccio
3	Algorithm can be retrained	Must	3	Code can retrain the algorithm	08-Feb	Michele Meo
4	Blade is tested multiple times	Must	3	Code can repeat the test	08-Feb	Marco Boccaccio
5	Algorithm is validated	Must	3	K-fold applied to training data	10-Feb	Nick Thomas
6	Algorithm is optimised	Must	3	Code automatically optimises algorithm	28-Mar	Nick Thomas
7	Application does not crash	Must	2	Error handling is implemented	05-Mar	Nick Thomas
8	Collect training data quickly	Must	2	Code can save tests quickly to CSV	05-Mar	Nick Thomas
9	Code can be used in the future	Wish	2	Code is created with best practices	10-Feb	Nick Thomas
10	Software is easy to use	Wish	2	GUI is simple	10-Feb	Nick Thomas
11	Clear pass/fail output	Wish	1	GUI clearly shows the test result	10-Feb	Nick Thomas

4.3 Selection of Programming Language

There was a wide choice of programming languages available, but MATLAB software was deemed to be the most appropriate for the project. Firstly, this was because MATLAB has powerful built-in toolboxes such as the Statistics and Machine Learning Toolbox, Signal Processing Toolbox and Parallel Computing Toolbox, which makes creating code easier as there are already fundamental functions available. Secondly, all mechanical engineers at the university have used MATLAB in previous years, ensuring that new features can be added easily.

As the application was created in MATLAB 2020b, that version or a newer was required to open and run the application. The code can be found in its entirety on GitHub.

4.4 Machine Learning Types

The aim of the project was to discern whether blades were damaged or undamaged. This could be achieved using different types of ML methods and particular algorithms, depending on the training data available. This aim is stage 1 of the description of the damage state of a system, which can be seen in Table 11 [61]. The different damage state descriptions can each be solved using different ML methods.

Table 11 – Description of the different damage states of a system [61]

State		Damage State Description
1	Existence	Is there damage in the system?
2	Location	Where is the damage in the system?
3	Type	What kind of damage is present?
4	Extent	How severe is the damage?
5	Prognosis	How much useful life remains?

The first method was to simply classify the unknown blade using training data which was labelled and contained results from undamaged and damaged blades. This method would be able to be expanded to identify the type and severity of the damage itself with enough training data, hence satisfying states 1 to 4 of Table 11 [61]. The problem was solved with supervised ML algorithms, as the training data had results for undamaged and damaged blades. These algorithms outputted the exact probability to which damage class the unknown blade belonged to.

The alternative method was to classify the blade using training data which only contained results from undamaged blades. This would only be able to recognise the existence of damage, but be unable to identify its type, thus satisfying states 1 to 2 of Table 11 [62]. This method required the use of unsupervised ML algorithms and was based on identifying clusters in the data [63]. These algorithms, on the other hand, only outputted if the unknown blade was in the undamaged cluster or not.

Supervised and unsupervised ML algorithms use the extracted features from the raw signal, such as the location of the natural frequencies, to create boundaries and patterns which can be used to classify unknown blades. The process of creating a supervised and unsupervised models is explained below, with the associated results discussed in Section 5.

4.5 Machine Learning – Supervised

There were two possible methods for supervised ML as discussed in the literature review, conventional ML and DL, the main difference is shown in Figure 29. These were both explored in the project as possible solutions, as each had their own benefits as well as negatives.

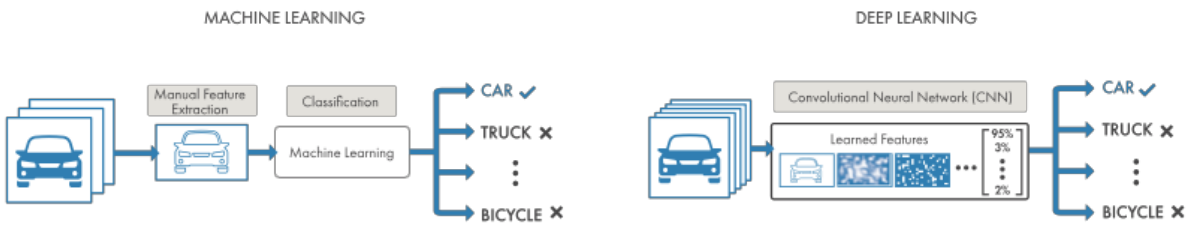


Figure 29 – Diagram showing the key difference between ML and DL [64]

4.5.1 Conventional Machine Learning

The conventional ML method requires the user to input the key features of the data. This is only practical if the user has domain knowledge of what these are, and then they can be easily selected. For this project, the feature selection was managed with the basic knowledge of the shift in natural frequency between undamaged and damaged blades. The peak selection was accomplished in the GUI, where the FFT amplitude response was plotted and could be easily used to select the peaks.

However, from the user's selection, not all the peaks would be at the natural frequencies of the blade, as some would be of the hammer and some reflections from the walls, see Figure 30 where the peaks above 30 – 45 kHz are likely to be reflections. To counter this, the features could be passed into a Principal Component Analysis (PCA) which reduces the dimensionality of the data, whilst keeping the features that actively change between blades [65]. However, this is best applied to large datasets, which inherently have a sizeable amount of variation, to reduce the longer training time whilst having a minimal effect on the accuracy.

Furthermore, the training time and optimisation of these algorithms for the training files were typically only a few minutes. This allowed the user to select a smaller number of peaks, view the accuracy of the model and, if required, gradually select more peaks. It can be seen from Figure 30 that the user would be highly confident with the six peaks displayed by the red circles. However, the remaining peaks, especially those between 30 – 45 kHz are much less distinguishable.

There is a wide variety of different algorithms as seen in the literature review, including a shallow neural network with one or two hidden layers. The GUI was designed to allow the user to select different algorithms, depending on which performed best.

The benefits of conventional ML were that the algorithms were very quick to train and optimise, and do not usually need a large dataset to perform well. However, a negative is that the user must manually select the key features of the data.

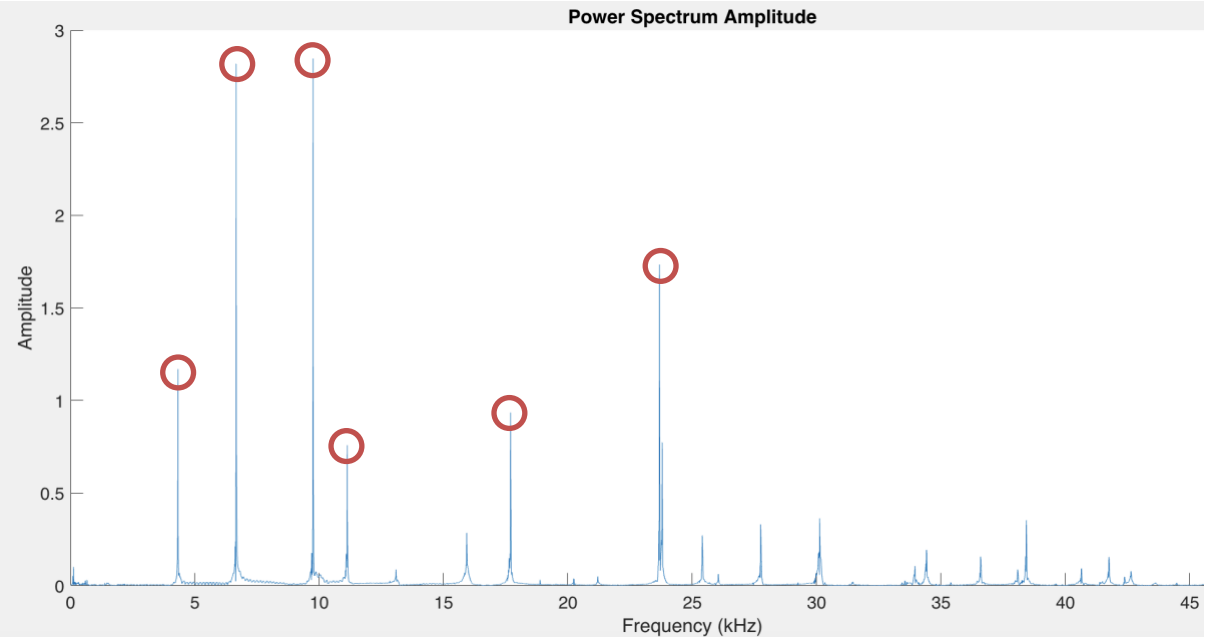


Figure 30 – FFT amplitude plot highlighting the key natural frequencies

4.5.2 Deep Learning

DL only requires the input of the raw data. It automatically finds the key features and therefore trains itself. DL is based on neural networks but instead of only using a couple of hidden layers, like the example in Figure 31, it uses hundreds of layers [64]. These layers are each responsible for extracting a specific feature from the image.

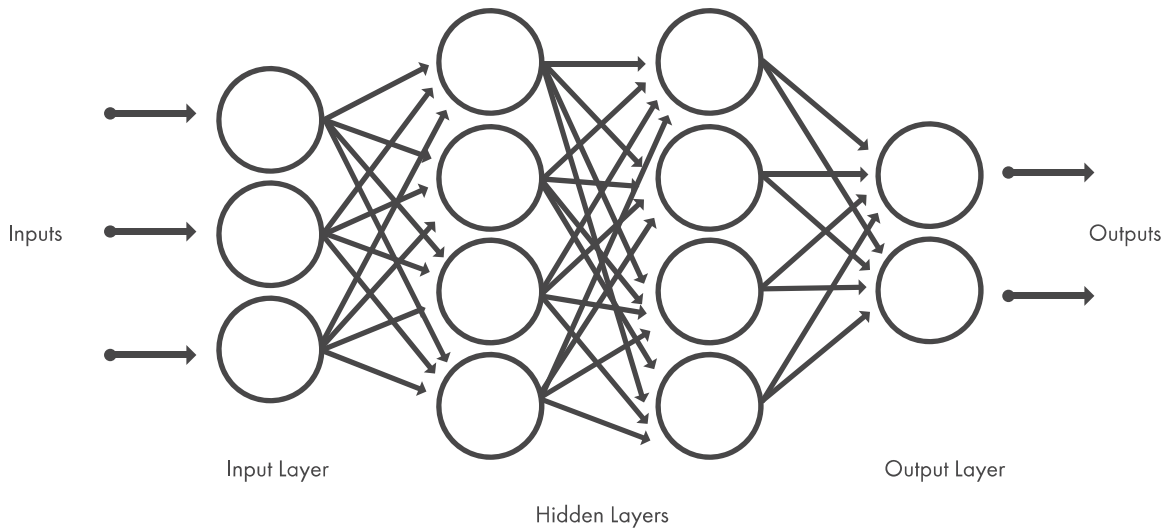


Figure 31 – Conventional shallow neural network layout diagram [64]

The most common DL network is a Convolutional Neural Network (CNN) which requires two-dimensional image data as inputs. The data obtained from the oscilloscope was time series and therefore only in one dimension. However, the data could be represented as a spectrogram, which is a visual representation of the spectrum of frequencies and their associated amplitude of a signal, as it varies with time [66]. A spectrogram is essentially an FFT amplitude plot which varies with time. An example spectrogram of a blade is shown in Figure 32.

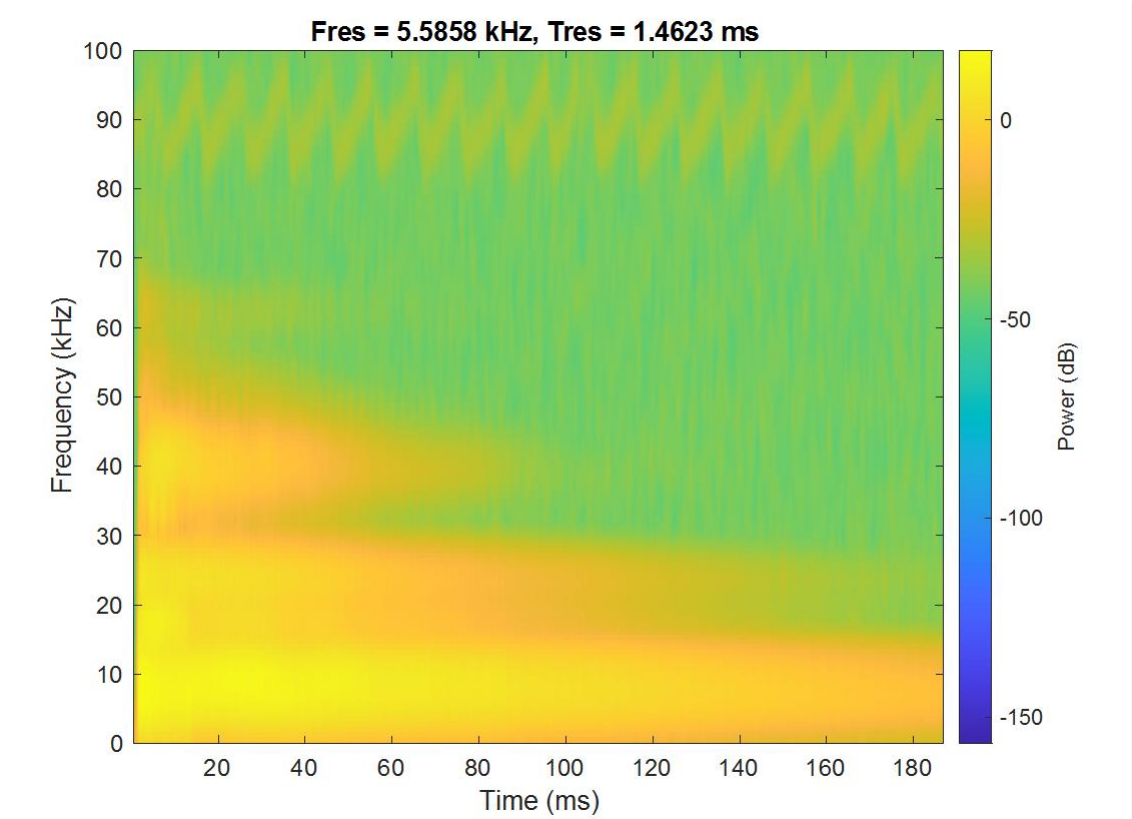


Figure 32 – Example spectrogram for a test blade

However, the time aspect of the spectrum must be the same between blades, therefore the Comma-Separated Values (CSV) files need to be standardised. This was achieved by removing the datapoints before impact to ensure that the impact event was at the same datapoint and that the number of datapoints after the impact event was the same. The axis, title and labels had to be removed, to leave solely the actual spectrum, which was then used in the DL model, shown in Figure 33.

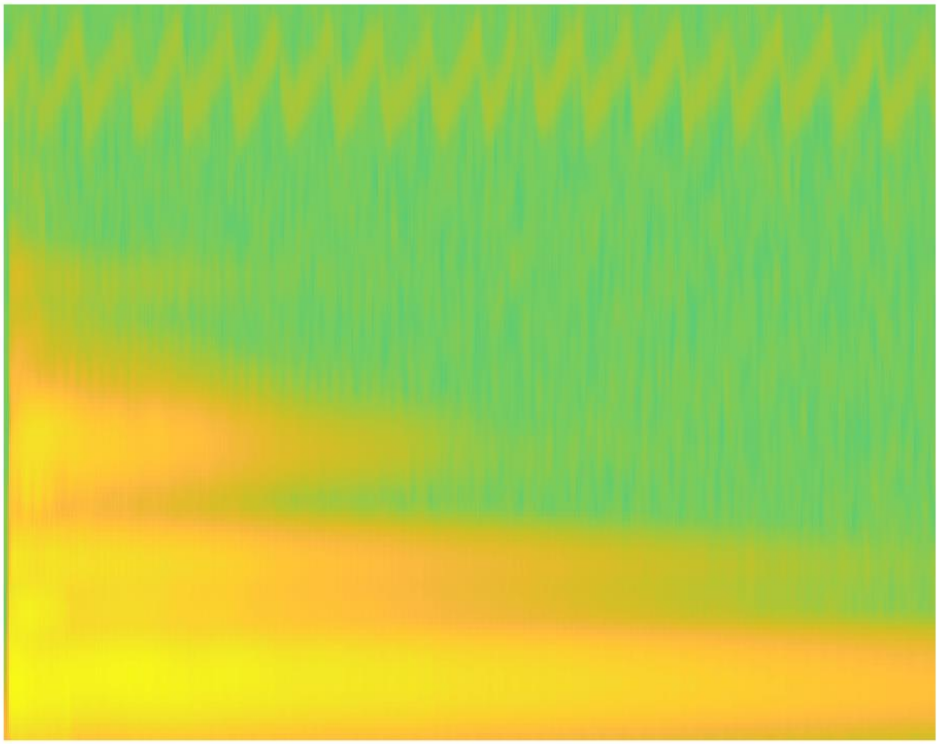


Figure 33 – Example of the spectrogram image inputted into the DL model

CNNs could be created from scratch, where the network architecture is designed layer by layer to achieve the required feature extraction performance. This was not implemented for the project because firstly, there needed to be a large, labelled dataset to train the network and secondly, the training, plus optimisation of parameters, would have taken days or weeks, even on powerful GPUs, therefore this was not selected for the project.

Instead, the project used transfer learning where a pre-trained model was fine-tuned. This was achieved by modifying several internal layers to match them to the requirements of the project. For example, only classifying two classes, undamaged and damaged blades, instead of thousands of classes [67].

The benefits of DL were that the user does not need to select the features themselves and the performance continues to improve as the size of the training dataset increases. Whereas this is not the case for conventional ML, which plateau at a certain level [64]. However, there were significant drawbacks associated with DL. One such negative was the requirement to use a large dataset to produce accurate predictions, which typically would involve hundreds to thousands of files. Furthermore, CNNs need substantial computing power to be trained and optimised, and even with this, the process takes a long time to complete.

4.6 Machine Learning – Unsupervised

In a similar way to supervised ML, unsupervised ML requires the user to input the key features of the data. This was completed in the same way as described in Section 4.5.

Unsupervised ML algorithms are generally used to fit clusters and find hidden patterns in datasets by using a measure of similarity [68]. These algorithms usually utilise a distance metric such as Euclidean distance, as shown in Equation 4, to fit points to clusters. The algorithms can also be used to compress data, or to identify and rank features. Again, there is a wide choice of different algorithms, as seen in the literature review. For this project, cluster analysis was the most suitable method to identify if the unknown blade fitted within the undamaged blade cluster or if it was an outlier and therefore damaged.

$$P(p_1, p_2) \quad , \quad Q(q_1, q_2)$$

$$d = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \tag{4}$$

Unlike supervised ML, a model is not created when using unsupervised ML, as only training data for the undamaged blades exists. However, for the project, the undamaged data was pre-processed before classification, which simply involved extracting the natural frequency points from the raw signal data and then saved to a file. This was primarily completed to speed up the classification process of unknown blades.

During classification, the training data was combined with the data of the unknown test blade and then the algorithms were used to fit clusters to the data. A single cluster is fitted if the test blade datapoint is within the undamaged datapoints, or two clusters if the blade is damaged, as seen in Figure 34 with the outlier point.

Figure 34 – Example of an outlier in the data producing a second cluster [69]

4.7 Model Validation and Optimisation

The validation of supervised ML models was an important step to ensure that the models were not only accurate but were also not overfitting the training data. Overfitting is when the model correctly classifies the training data but is unable to generalise new data.

The conventional ML models were validated during their training by using leave-one-out k-fold cross validation. A fold is a group of observations and in the leave-one-out case, each fold contains a single observation. The leave-one-out case operates by leaving a single fold out of the training which is used for validation. This process is repeated until every fold has been used once as the validation data, which in turn guarantees that the model is validated and generalised, ensuring that it can classify unseen blades. The DL models also have a validation step, where a portion of the data is withheld from the training and is tested against the model to ensure that it is not being overfitted.

The conventional ML models were also automatically optimised using Bayesian optimisation. This is an algorithm that attempts to minimize a scalar objective function f(x) for x in a bounded domain [70]. The algorithm modifies all the valid hyperparameters to identify values which produce the lowest loss. The loss indicates how many misclassifications the model made of the training data. This step is computationally expensive, as the model is fitted many times to ensure the optimum model has been created.

Conversely, DL models do not have hyperparameters; instead, they only have training parameters where the optimisation of the parameters is less important, as they mainly control the training speed. These parameters are unable to be automatically optimised during the training and therefore should be done beforehand. A MATLAB Experiment Manager test was set up which could perform this optimisation. However, this was not run due to the limited processing power available and the insufficient quantity of training data.

It was not possible to optimise the unsupervised ML methods because no models were created, as the algorithms were simply used for a single outlier identification for each unknown test blade.

4.8 Software Architecture

The application was built from a GUI and therefore an event-driven architecture was used where sub-functions are called, depending on what event has been triggered by the user interface, for example, to begin a classification test. These sub-functions then accomplished tasks and reported their progress back to the GUI, with any errors. The GUI was split into four tabs, where each one had a key responsibility and are explored in detail in Section 5.

4.9 The Main Functions

The application was designed to use sub-functions, which were kept in separate MATLAB .m files from the GUI for modularity. These sub-functions were designed to be as modular as possible, to make sure that the code could be easily used without the application, for in-depth data analysis as an example.

There were two tiers of sub-functions: those which contained logic to call the base functions and the base functions themselves. Examples of the logic sub-functions are **RunSingleTest.m** and **ImpactTest.m** found in Appendix A and Appendix B respectively. Figure 35 shows the code structure for both logic sub-functions. The base level functions were responsible for individual tasks, such as pre-process the raw data or classify the test blade. These base functions are explained in Appendices C to J where the source code is also presented.

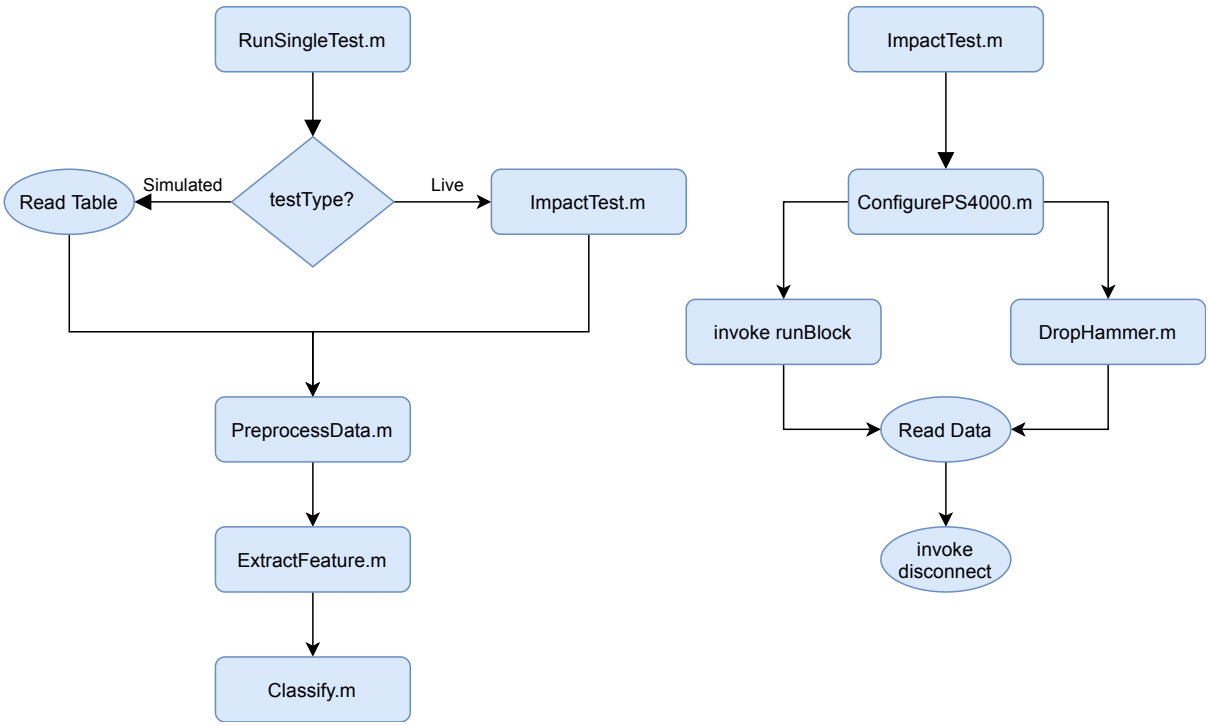


Figure 35 – Code structure for the two top level logic functions

4.10 Solution Specification

Table 12 summaries the results of the MATLAB application against the requirements list. The application meets all 11 requirements, concluding that it was successful. Many comments were added to the code to enable future engineers to understand the code quickly and to add capabilities.

Table 12 – Solution specifications compared to the software requirements list

	Requirements	Success criteria	Actual	Success/ fail
1	Test is autonomous	One button to run test	Yes	✓
2	Control the test rig	Code can interface with the hardware	Yes	✓
3	Algorithm can be retrained	Code can retrain the algorithm	Yes	✓
4	Blade is tested multiple times	Code can repeat the test	Yes	✓
5	Algorithm is validated	K-fold applied to training data	Yes	✓
6	Algorithm is optimised	Code automatically optimises algorithm	Yes	✓
7	Application does not crash	Error handling is implemented	Yes	✓
8	Collect training data quickly	Code can save tests quickly to CSV	Yes	✓
9	Code can be used in the future	Code is created with best practices	Yes	✓
10	Software is easy to use	GUI is simple	Yes	✓
11	Clear pass/fail output	GUI clearly shows the test result	Yes	✓

5 Results and Discussion

This section reviews and discusses the key results from the project. This includes the GUI, the test rig and communication flow, the experimental procedure and the accuracy of the different ML models. The errors, uncertainty and limitations of the test rig and MATLAB application are also discussed.

5.1 Graphical User Interface

The project owner constructed the GUI from scratch and therefore this section covers the GUI itself, as it is a key result of the project. The code of the created GUI can be found in Appendix L.

5.1.1 Main Pass/Fail Application Tab

The *Main Pass/Fail Application* tab was responsible for the classification of unknown blades using the trained ML model. The tab can perform classification of a simulated CSV file from a previous test or from a live test. It plots the raw data and FFT amplitude on the right, with information about the test setup, progress, and results on the left. The test comprises of a set of five separate impacts on the blade to produce an average for the classification, ensuring the blade is categorised correctly.
The GUI of the tab was split into five panels, as shown in Figure 36.

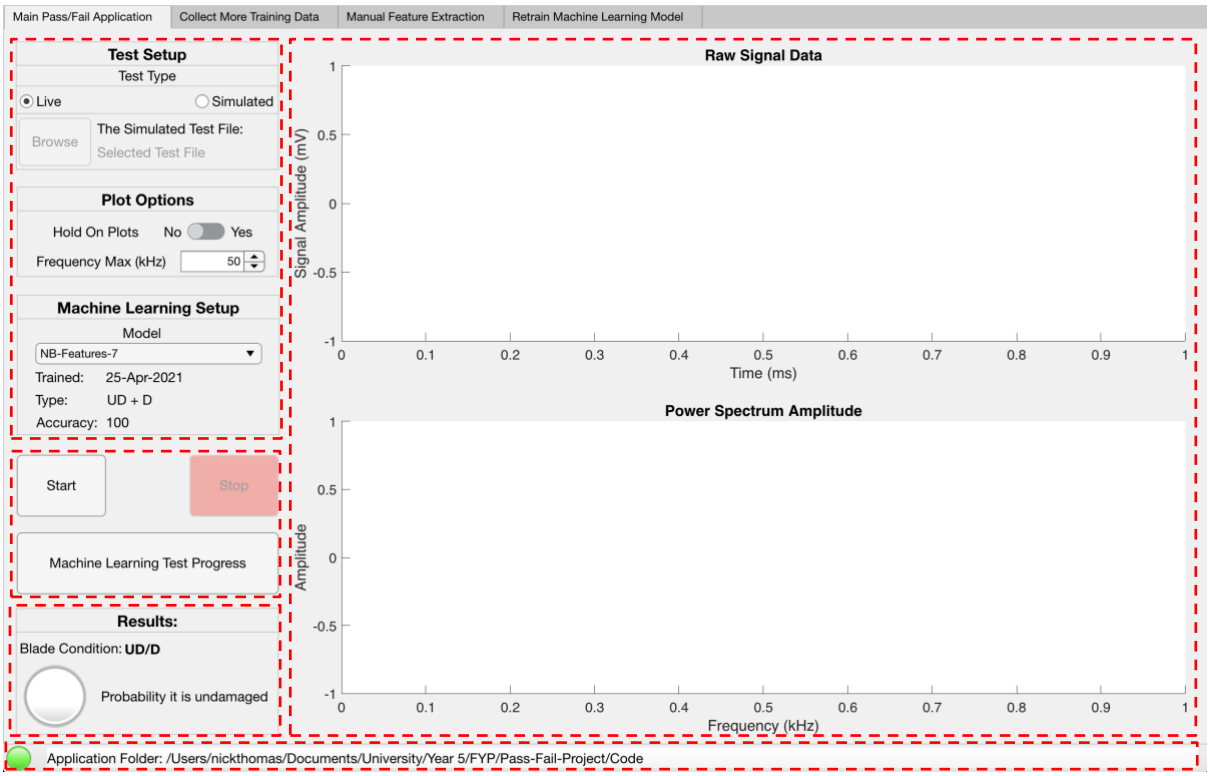


Figure 36 – Main Pass/Fail Application tab split into five panels by red boxes

Figure 37 shows the test setup, plot options and ML setup for the *Main Pass/Fail Application*. Figure 38 shows the test controls with two buttons and a progress bar. Figure 39 shows the test results for the blade.

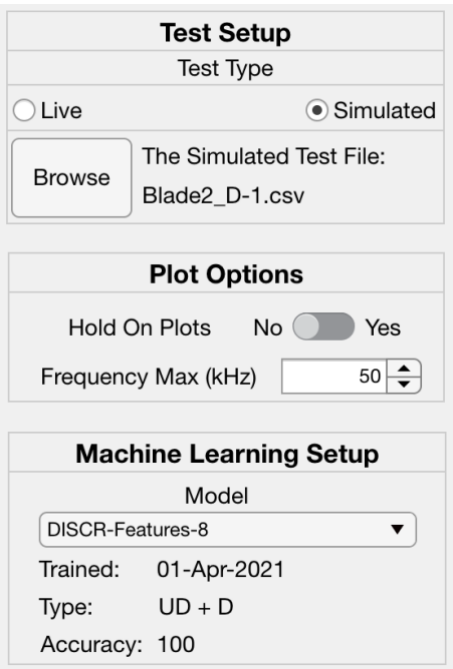


Figure 37 – Tab 1 panel 1: Test setup

The radio button is used to select the test type. If it is set to *Live,* then the *Browse* button is greyed out and disabled.
Otherwise, it is enabled to allow selection of the simulated test file. Only .csv files can be accepted by the *Browse* button and if an invalid type has been selected then the *Start* button remains greyed out.

The x axis can be adjusted depending on the blade's response and the detail required. The test performs multiple impacts on the blade to gain an average and therefore the previous results can be kept on the plots if required.

The drop down menu allows the required model to be selected, whilst also showing key pieces of information.

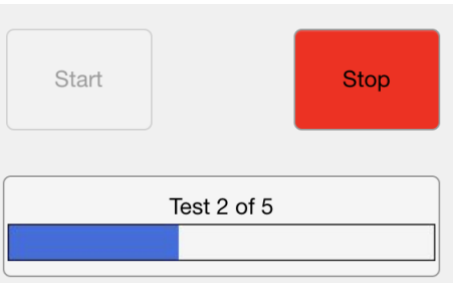


Figure 38 – Tab 1 panel 2: Test controls

The *Start* button is only enabled when there is either an acceptable CSV file selected, or the test type is set to be *Live*. After the *Start* button has been clicked, it greys out and is disabled. The *Stop* button is enabled, allowing the test to be stopped if there is a problem.

The progress bar shows the progress of the test and how many more separate impacts are required to be performed.

Figure 39 – Tab 1 panel 3: Test results

The *Blade Condition* button changes colour depending on the results of the test. Red shows the blade is damaged, green shows the blade is undamaged and white shows the condition is unknown, which advises performing a repeat test.

Figure 40 shows the plots of the data from the test. The first plot shows the raw signal collected. The second plot shows the FFT amplitude response from the raw data, with the red triangles highlighting the natural frequencies used in the ML model.

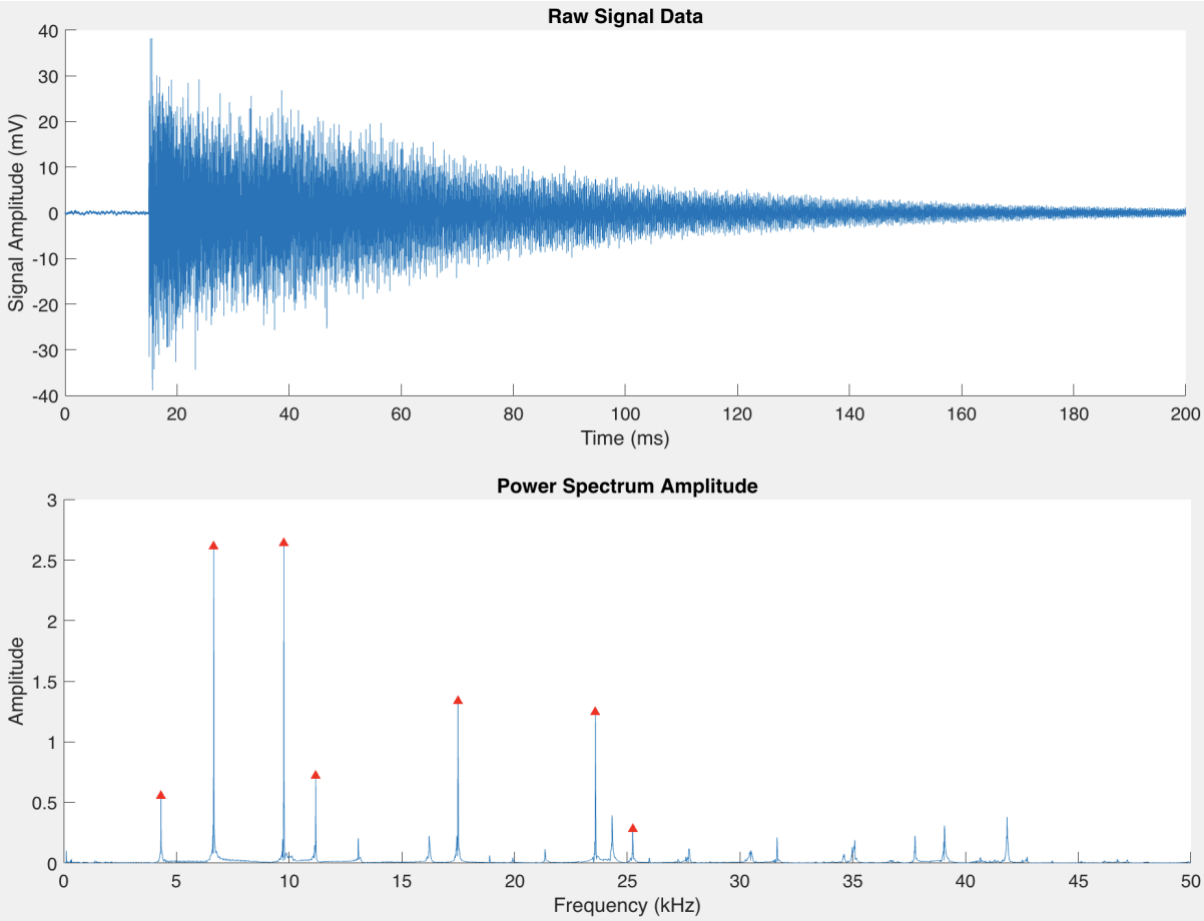


Figure 40 – Tab 1 panel 4: Raw signal and FFT amplitude plots

Figure 41 shows the status bar, which is at the bottom of each tab. This bar informs the user of the condition of the application; whether there are any errors and if so, where the error originated. The application was designed to be able to manage errors and to produce dialog pop-ups with information on what errors have occurred and how to fix them. The status bar is present in every tab and is not discussed further.

Figure 41 – Tab 1 panel 5: Status bar

5.1.2 Collect More Training Data Tab

The *Collect More Training Data* tab was responsible for quickly collecting more training data of the labelled blades. The tab provides the option to select the location where files can be saved, the name of the files and the number of tests to be conducted. The tab plots the signal of each test and updates the user on the progress of the collection.
The GUI of the tab was split into three panels, as shown in Figure 42.

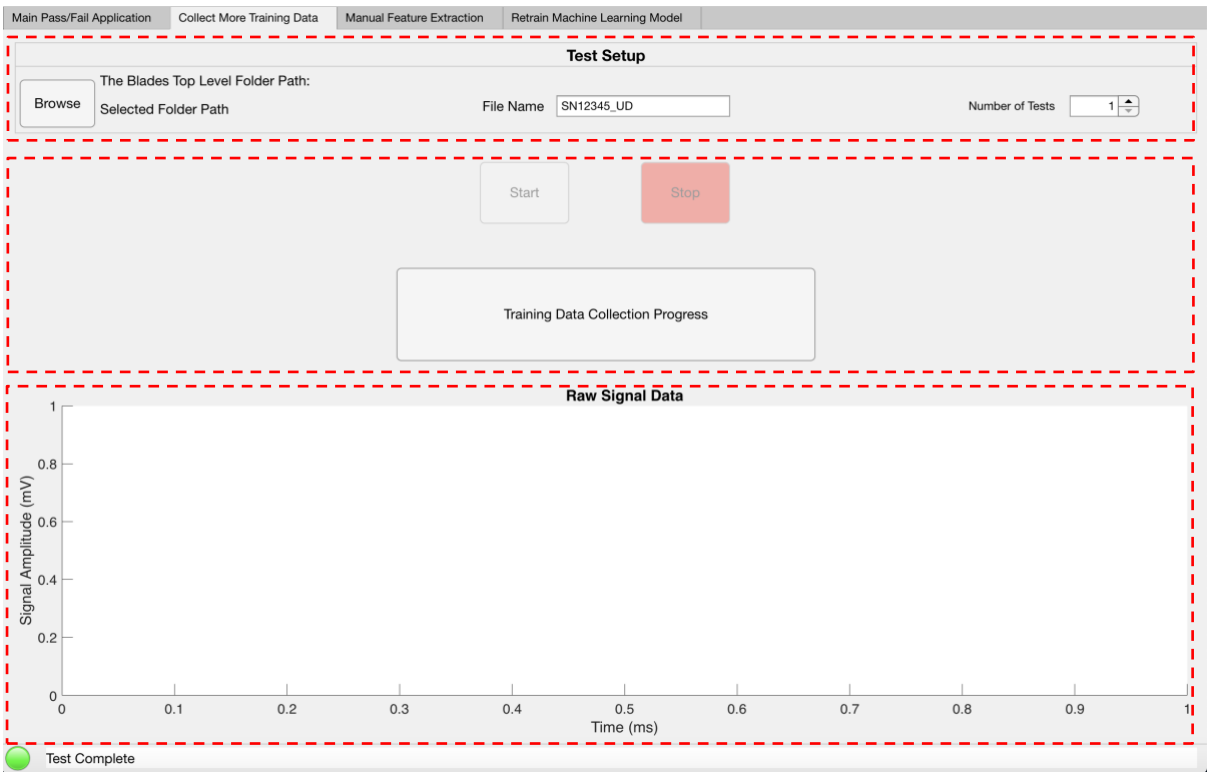


Figure 42 – Collect More Training Data tab split into three panels by red boxes

Figure 43 shows the test setup for the *Collect More Training Data* tab. The *Browse* button functions in the same way as the *Main Pass/Fail Application* tab, but instead selects a folder path for where the new training data should be saved. The file name can be entered into the *File Name* box in the format of the serial number, followed by the known damage condition of the blade as seen in Figure 43. In addition, the number of tests needed can be entered. The application searches the selected folder for previous folders and files with the same file name and ensures that none are overwritten, by increasing the index value of the file name.

Figure 43 – Tab 2 panel 1: Test setup

Figure 44 shows the test controls with two buttons and a progress bar. The *Start* button is only enabled when a valid folder is selected. After the *Start* button has been clicked, it greys out. The *Stop* button is then enabled allowing the test to be stopped if there is a problem, or if the user wishes to collect a smaller number of tests. The progress bar shows the progress of the collection and how many more tests need to be performed.

Figure 44 – Tab 2 panel 2: Test controls

Figure 45 shows the raw signal collected from each test. The data is then saved to a CSV file with the name of the file and the folder specified at the top of the tab. The data is processed to remove any datapoints before the impact and is then checked to make sure that it does not contain any errors.

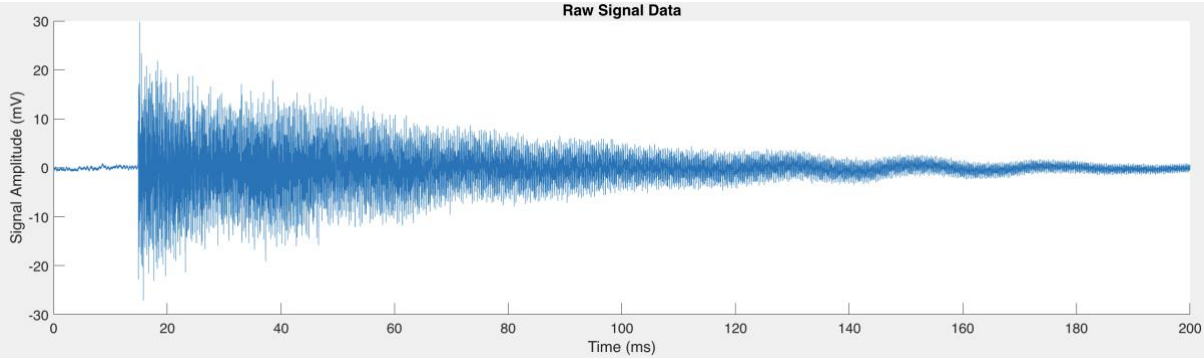


Figure 45 – Tab 2 panel 3: Raw signal plot

5.1.3 Manual Feature Extraction Tab

The *Manual Feature Extraction* tab was responsible for enabling the user to manually select the estimated location of the natural frequency peaks. This was required for all ML algorithms, apart from DL.
The GUI of the tab was split into three main panels, as shown in Figure 46.

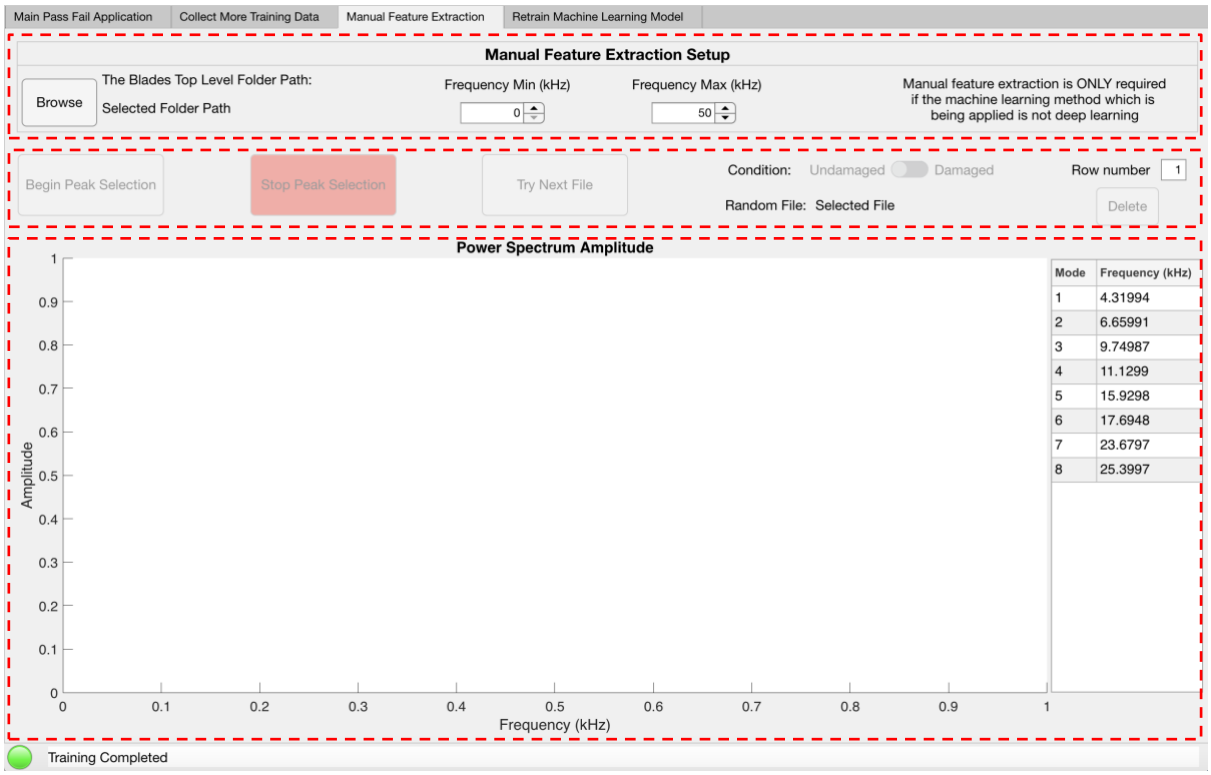


Figure 46 – Manual Feature Extraction tab split into three panels by red boxes

Figure 47 shows the test setup for the *Manual Feature Extraction* tab. The *Browse* button functions in the same manner as the *Collect More Training Data* tab. The training data for the damaged and undamaged blades is kept in the folder location. Furthermore, the x axis limits of the plot can be adjusted.

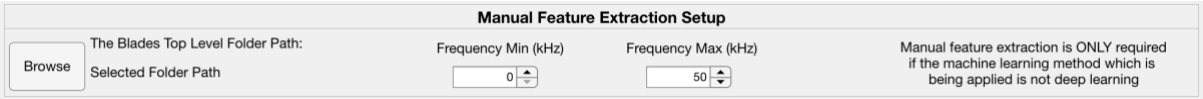


Figure 47 – Tab 3 panel 1: Tab setup

Figure 48 shows the test controls with the *Begin Peak Selection* and *Stop Peak Selection* buttons operating in the same way as the previous buttons in Figure 44. The *Try Next File* button and the *Condition* switch can be used to select a different file to view. The *Delete* button can be used to delete any natural frequency peak values which were previously selected.

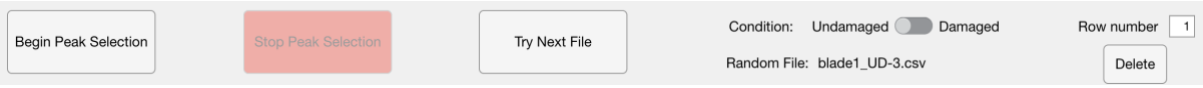


Figure 48 – Tab 3 panel 2: Tab controls

Figure 49 shows the FFT amplitude plot used for peak selection. The list of previous peaks is shown in the table on the right and the peaks are highlighted by the dashed orange lines on the plot. The plot becomes interactive once the *Start Peak Selection* button has been selected and when the user hovers the mouse at a peak, the peak is automatically selected by adding the dashed line and updating the table in ascending order.

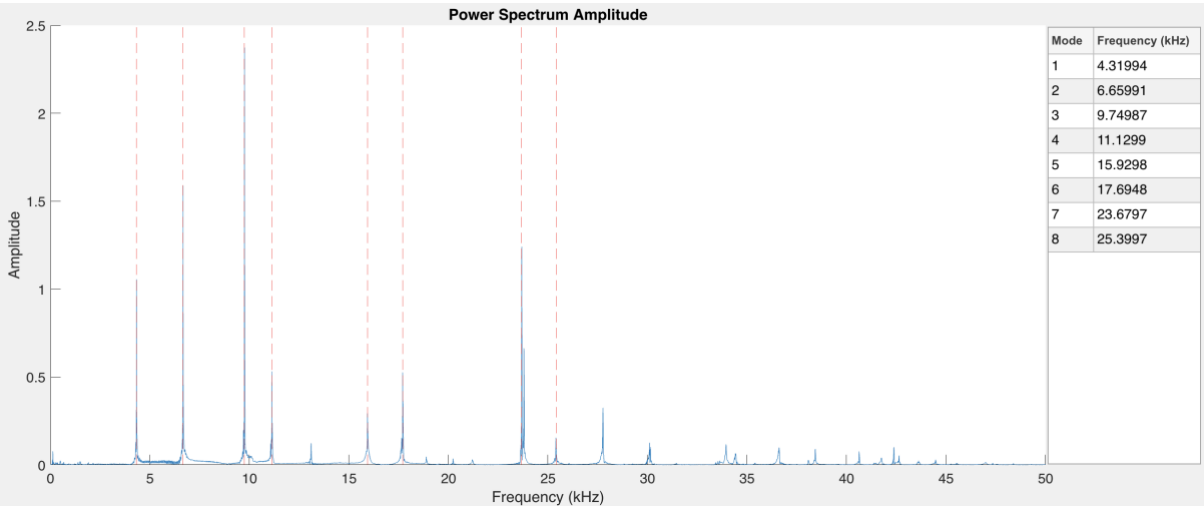


Figure 49 – Tab 3 panel 3: Interactive FFT amplitude plot

This peak selection was designed to be an iterative process, where the user selects a peak from one CSV file and then uses the buttons to change to another random undamaged or damaged CSV file to identify if the peak remained. Some peaks were false natural frequencies caused from reflections, which could be detected by viewing the other files. The user could consequently delete the false peak and select a different one.

5.1.4 Retrain Machine Learning Model Tab

The *Retrain Machine Learning Model* tab was responsible for training all the ML models. The tab displays the previously trained models, options to train the next model and results from the trained model.
The GUI of the tab was split into three main panels, as shown in Figure 50.

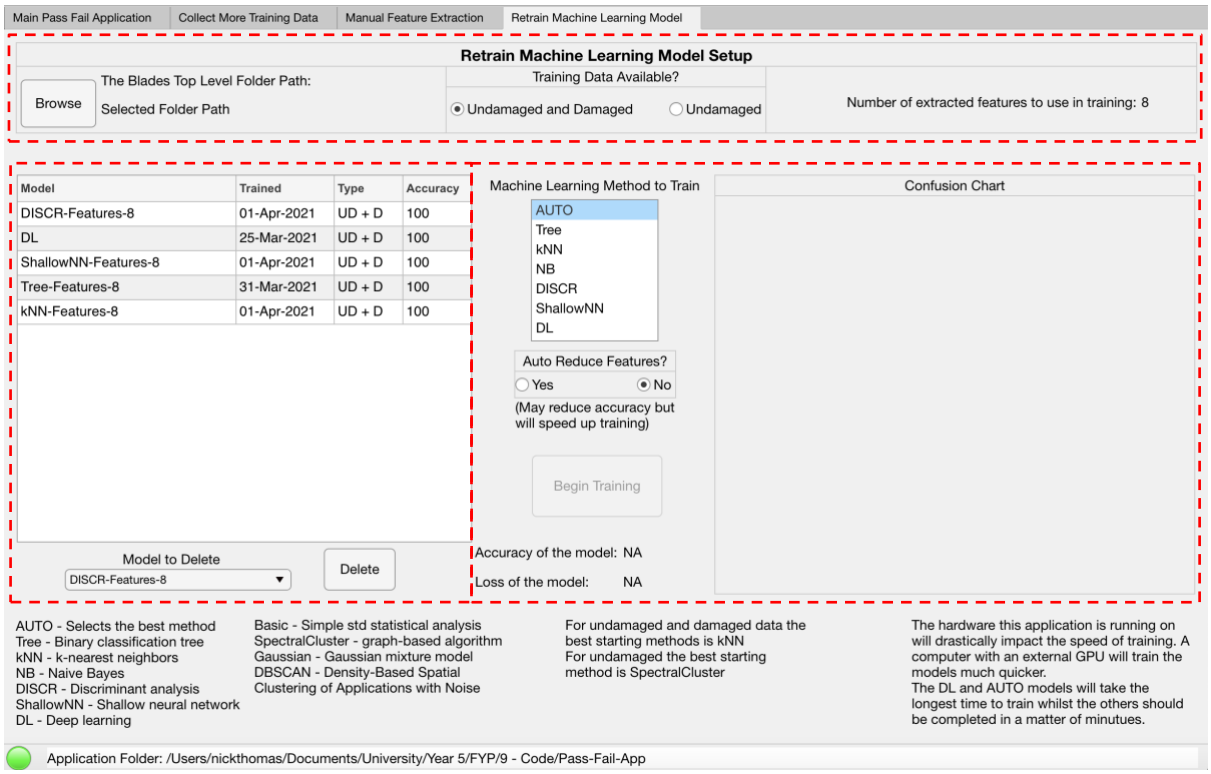


Figure 50 – Retrain Machine Learning Model tab split into three panels by red boxes

Figure 51 shows the test setup for the *Retrain Machine Learning Model* tab. The *Browse* button functions in the same way as the *Manual Feature Extraction* tab. The radio button is used to select the type of training data available: undamaged and damaged for supervised ML models and damaged only for unsupervised ML models.

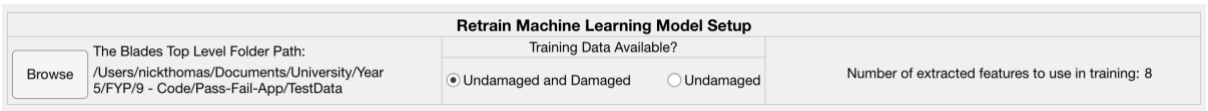


Figure 51 – Tab 4 panel 1: Tab setup

Figure 52 shows the table of previously trained ML models with information about the accuracy, type, when it was trained, and the number of features used. It also includes a button to delete the previous models from a populated drop-down menu.

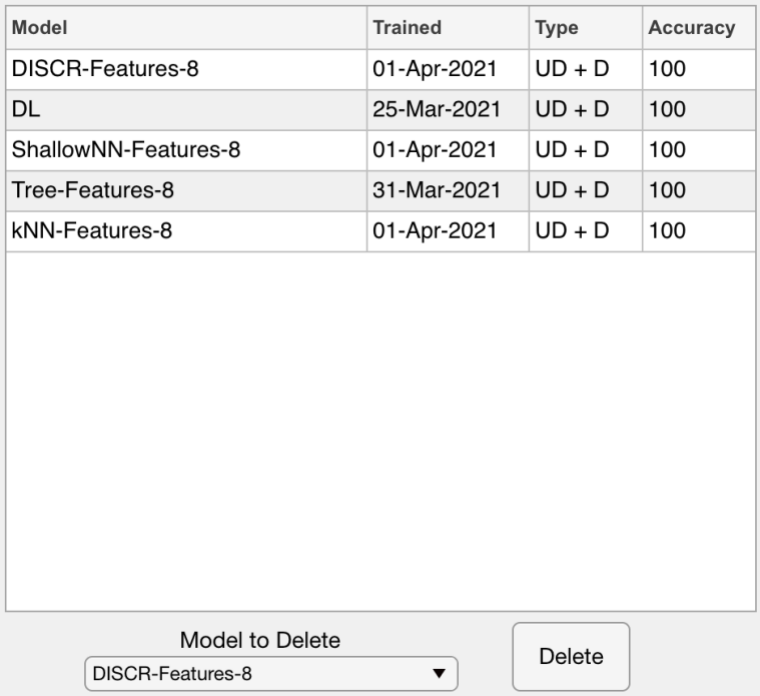


Figure 52 – Tab 4 panel 2: Trained model table

Figure 53 shows the training selection options and results from the trained model. The ML method list is populated, depending on which radio button is selected, as seen in Figure 51. The number of features can be reduced by using the *Auto Reduce Features* radio button, which performs PCA. When a model is trained, the data is split into training, validation and testing data. The results from the trained model are the accuracy of the testing data, which is also presented in a confusion chart. The loss of the model is also presented.

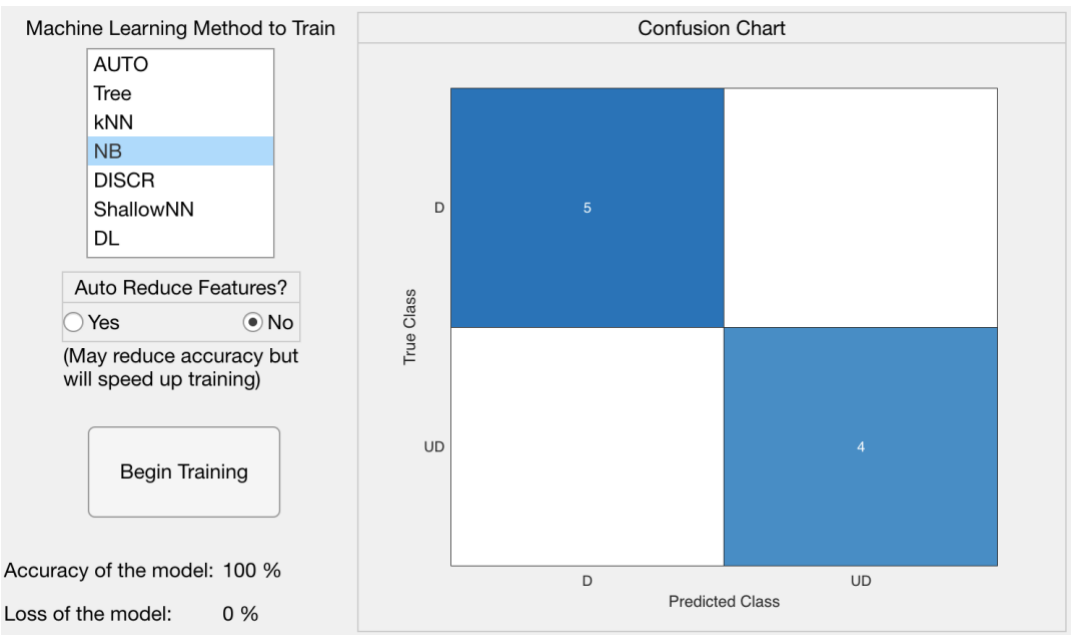


Figure 53 – Tab 4 panel 3: Training selection and results

5.2 Test Rig and Communications Flow

The main pieces of hardware required for the test rig are presented in Table 13.

Table 13 – Main pieces of hardware required for the new test rig [71], [72]

Hardware name	Image
Oscilloscope Picoscope 4424 Kit	
Microphone NCG100-D25-P76	

Figure 54 shows a picture of the new rig with the key pieces of hardware and equipment labelled. The new rig has the same lever mechanism as the proof of concept rig but is actuated using a more repeatable method with a stepper motor. Figure 55 shows the internal view of the enclosure with the impact hammer and test blade.

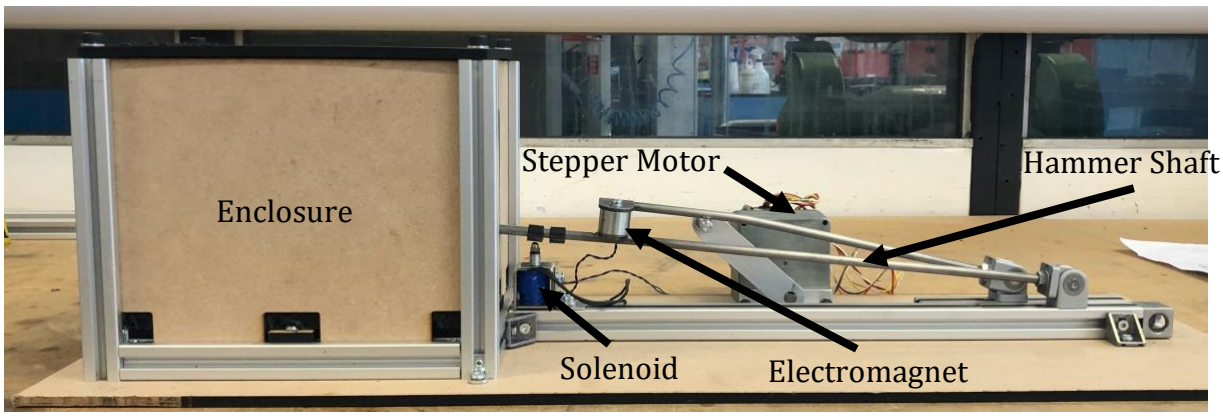


Figure 54 – Side view of the new test rig highlighting the key components

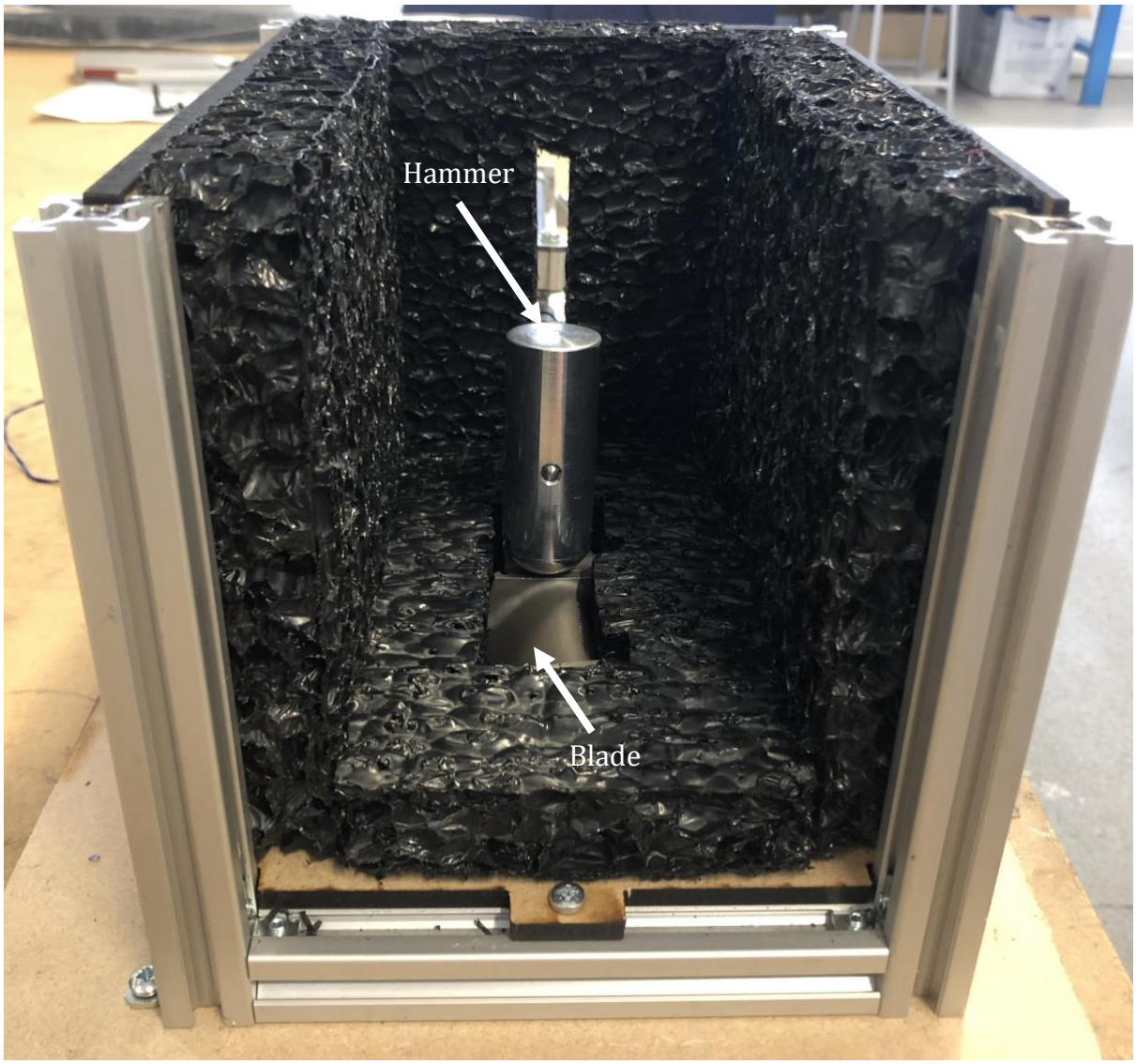


Figure 55 – Front view of the new test rig with the front and top panels removed

Figure 56 demonstrates the flow of communication for the test rig. MATLAB connects directly to the Arduino and to the oscilloscope. The oscilloscope is a multiple channel device which allows several microphones to be connected at the same time.

To begin a test, the user simply clicks *Start* on the GUI and MATLAB automatically performs the rest of the communication. When the button is clicked, MATLAB sends a signal simultaneously to the Arduino to drop the hammer and to the oscilloscope to begin recording microphone data. The microphone data is then read by MATLAB and is processed to extract the natural frequencies. Depending on what function is being performed, the data is either passed into the ML algorithm or saved to a CSV file. This is automatically repeated either five times to achieve an average classification for the blade, or replicated for the number of repeated tests specified.

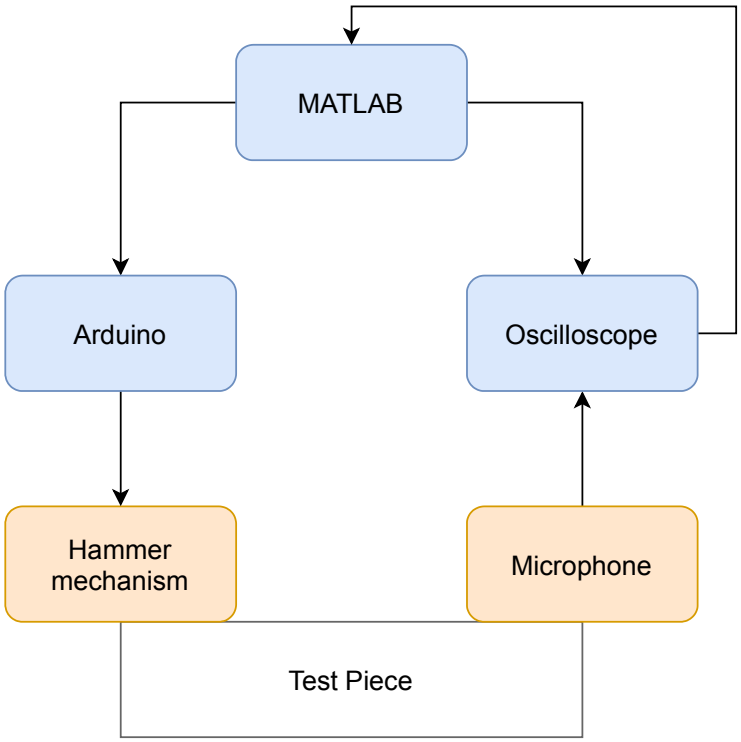


Figure 56 – Diagram showing the flow of communication for the test rig

5.3 Experimental Procedure

The process of conducting a test was designed to be as simple as possible, with a large amount of the test performed automatically from the MATLAB application. The process can be broken down into four distinct steps. However, if the required model has already been trained, then steps one to three can be omitted. A simplified step by step user documentation has been created and can be found in Appendix M.

1. Use the *Collect More Training Data* tab of the GUI to run tests on a large quantity of blades.
2. Use the *Manual Feature Extraction* tab of the GUI to select the key features of the data. This step is only required if a DL model is not being used.
3. Train the most appropriate ML model using the *Retrain Machine Learning Model* tab.
4. Select the chosen ML model and then run the test with the unknown blade from the *Main Pass/Fail Application* tab.

5.4 Results and Analysis of the Machine Learning Algorithms

This section analyses the results outputted from the ML models. Due to the Covid-19 restrictions, the project owner was unable to conduct the tests and a technician had to collect the data on his behalf. Consequently, only three blades were tested, resulting in a very limited dataset, with a small number of repeats for each blade. In addition, the supervisor did not have any blades that were completely undamaged and therefore the models were trained and tested with blade one being 'unknown', blade two being 'damaged' and blade three being called 'undamaged' which appeared the least damaged in terms of the natural frequencies. Furthermore, the proof of concept rig had been damaged and therefore the test data was obtained using a roller ball impact outside an enclosure, see Figure 57. This resulted in responses which would be slightly different to the ones obtained by the new test rig.

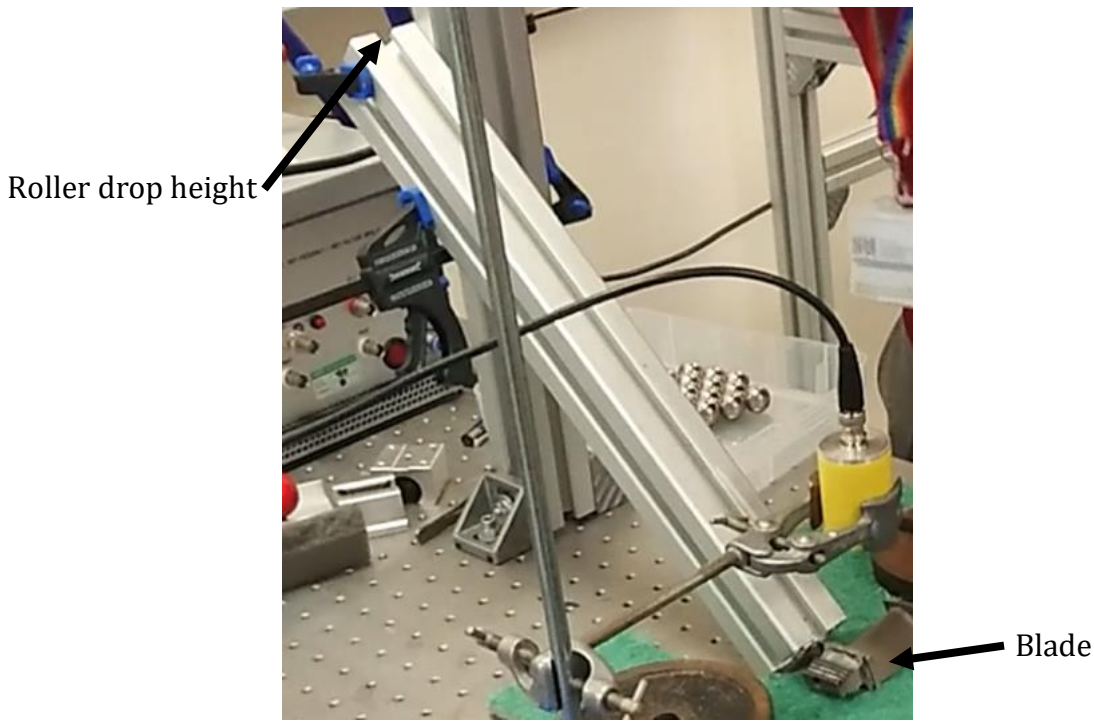


Figure 57 – Roller ball impact rig used to collect the ML training data

As a result, the investigations and subsequently outcomes may not be representative when a full dataset obtained from the new test rig is used. The investigations were to prove that the test rig and MATLAB application were functioning reliably and could be used in the future. The investigations should be repeated when a dataset is created which has enough variation, for example, with five undamaged blades and 15 blades with varying degrees of damage, each with 50 repeats, producing a dataset of 1000 files.

5.4.1 Repeatability and Reproducibility

To ascertain the exact repeatability and reproducibility of the whole rig, a Gage Repeatability & Reproducibility (Gage R&R) was necessary. A Gage R&R is a method to define the amount of variation in the measurement data due to the measurement system. In the case of this project, it was the impact hammer mechanism, microphone and blade location [73]. However, since the test data did not come from a rig and was obtained via a roller ball impact, only the repeatability of the GUI and the feature extraction from the raw signal was analysed by calculating the mean and standard deviation for the main natural frequencies. For the remainder of the analysis, the seven most appropriate natural frequencies for each CSV file are used and can be seen in Figure 58.

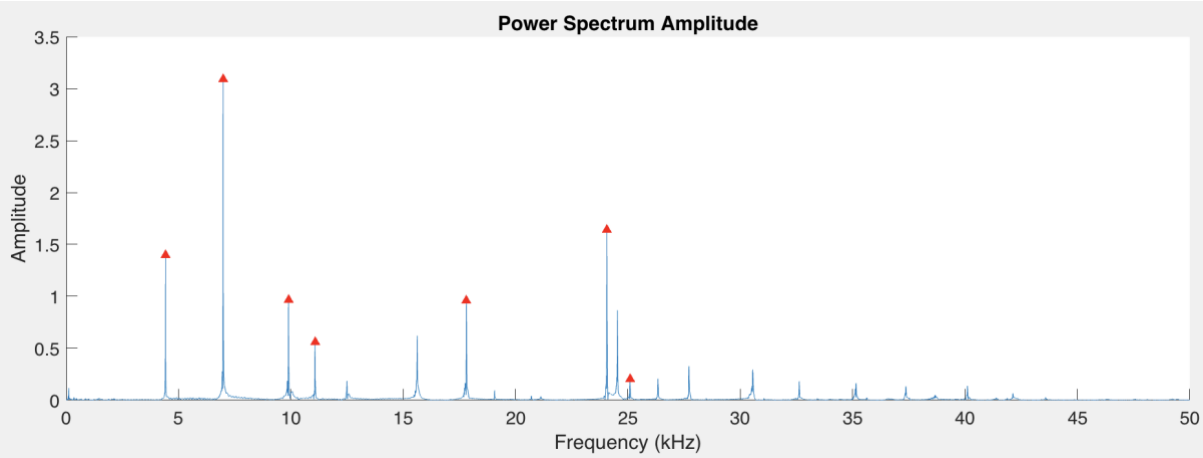


Figure 58 – The seven most appropriate natural frequencies used in the analysis

Table 14 presents the mean and standard deviations for the three different blades. The repeatability of the data was high, as several standard deviations were either zero or of a miniscule value. The results were consistent to one datapoint resolution, namely 5 Hz, due to the sampling frequency of the oscilloscope.

Table 14 – Mean and standard deviations of natural frequencies for the test blades

Blade	One – Unknown (Hz)		Two – Damaged (Hz)		Three – Undamaged (Hz)	
Natural Frequency	Mean	Std	Mean	Std	Mean	Std
1	4,320	0	4,315	0	4,425	0
2	6,660	0	6,655	0	6,990	0
3	9,750	0	9,760	0	9,896	1.85
4	11,130	0	11,175	0.88	11,081	2.34
5	17,695	0	17,485	0	17,812	2.50
6	23,680	0	23,585	0	24,065	0
7	25,400	2.58	25,240	0	25,090	0

5.4.2 Supervised Machine Learning Results

To compare the different supervised ML models, two stages were used. Firstly, the analysis of training the model itself and the results associated with that process, and secondly, the analysis of how the model performed when classifying a previously unseen blade. All benchmarking tests were conducted on a 2017 MacBook Pro with i5 7360U CPU and integrated Intel Iris Plus Graphics 640 GPU.

5.4.2.1 Comparison of the Training of Various Machine Learning Models

The training times, accuracy of test data and loss of training data were used to analyse the different models. These values are presented in Table 15 for each ML algorithm with the seven selected features, as well as the PCA version. The dataset used for training was small with 30 damaged and 30 undamaged CSV files for reference. There were two files from each blade that were reserved for the known classification test.

Table 15 – Comparison of the training of supervised ML models

Algorithm	Number of Features	Training Time (seconds)	Accuracy (%)	Loss (%)
Binary classification tree	7	48	100	0
Binary classification tree PCA	1	36	100	0
k-nearest neighbours	7	48	100	0
k-nearest neighbours PCA	1	54	100	0
Naïve Bayes	7	70	100	0
Naïve Bayes PCA	1	37	100	0
Discriminant analysis	7	31	100	0
Discriminant analysis PCA	1	27	100	0
Shallow neural network	7	15	100	0
Shallow neural network PCA	1	14	100	0
Deep learning – GoogLeNet	-	4 minutes	39.5	76.5
Deep learning – ResNet-50	-	7 minutes	100	0.006
Deep learning – ResNet-101	-	12 minutes	100	0.0003

The convention ML models were all trained and optimised in approximately one minute, with the shallow neural network being the quickest at approximately 15 seconds. The neural network was created to have a single hidden layer with 15 neurons and therefore it was simple. The PCA variants of the models generally had a shorter training time by a few seconds, with the optimised number of features equalling one instead of seven. The use of PCA is not recommended for small datasets with little variation as the advantages gained are limited. The threshold of the PCA analysis was configured so that the reduced number of features encapsulated at least 97.5% of the total variance.

The DL models took much longer to train, at 4 minutes for the simple GoogLeNet, 7 minutes for the ResNet-50 and 12 minutes for ResNet-101. The dataset of 60 CSV files for DL was small; however, it is known that the training time would increase significantly with the correctly sized dataset.

The accuracy was calculated from the testing data, which amounted to 15% of the data collected. The accuracy was seen to be 100% for all models, apart from the DL GoogLeNet which had a much lower accuracy value. This was because GoogLeNet is a relatively simple DL model with only 22 layers compared to ResNet-50 and ResNet-101, which respectively have 50 and 101 layers. In practice, the accuracy would not be this high, as the models were only trained with two blades, which both had a very low standard deviation for the frequency measurements. In essence, this signified that the models were trained with two observations, making them 100% accurate. The loss for all the models was zero or very close to zero, apart from GoogLeNet. This was again due to the nature of the training data containing only two blades and the simple layout of GoogLeNet.

5.4.2.2 Comparison of the Classification of Blades

The models were then tested to classify blades which were not used in the training of the models. Three tests were performed; firstly, classifying the remaining repeats of blade three, secondly, classifying the remaining repeats of blade two; both blades had a known damage condition and thirdly, classifying the unknown blade which was not used in training the models. The results are presented in Table 16 where D is damaged and UD is undamaged.

Table 16 – Comparison of the classification using the supervised ML models

Algorithm	Blade 3 Class	Blade 2 Class	Blade 1 Class and probability of it being undamaged in %	
Binary classification tree	UD	D	D	0
Binary classification tree PCA	UD	D	D	0
k-nearest neighbours	UD	D	D	0
k-nearest neighbours PCA	UD	D	D	0
Naïve Bayes	UD	D	D	0
Naïve Bayes PCA	UD	D	D	0
Discriminant analysis	UD	D	D	0
Discriminant analysis PCA	UD	D	D	0
Shallow neural network	UD	D	D	0
Shallow neural network PCA	UD	D	D	0
Deep learning – GoogLeNet	UD	D	D	20
Deep learning – ResNet-50	UD	D	UD	86
Deep learning – ResNet-101	UD	D	D / UD	20 / 91

The results from tests one and two were that all models correctly identified the known blades. This was expected, as the test files were similar to the ones used to train the models. The third test produced mixed results with all conventional ML models and the DL GoogLeNet model classifying the blade as damaged. The probability that the blade was undamaged was low at 0% and 20%. However, due to the low accuracy and loss of the trained GoogLeNet model from Table 15, the classification was deemed to be inaccurate.

The ResNet models produced very interesting results with the 50 layer classifying the blade undamaged with an 86% probability, whilst the 101 layer classified it damaged for one repeat and then undamaged for the second repeat. These results clearly support the literature stating that DL models are only accurate when a large dataset is used to train them. For this project, the dataset used was too small and therefore the model does not adequately detect features in new data.

5.4.3 Unsupervised Machine Learning Results

The unsupervised ML models were not trained and therefore they could only be compared on their ability to correctly classify previously unseen blades. The same three tests were performed as Section 5.4.2.2, with the results shown in Table 17. PCA was not applied to the unsupervised models because all the variations in the data should be used when determining if the blade was an outlier.

Table 17 – Comparison of the classification using the unsupervised ML models

Algorithm	Blade 3 Class	Blade 2 Class	Blade 1 Class
Basic statistics	UD	D	D
Density-based spatial clustering of applications with noise	UD	D	D
Gaussian mixture	UD	D	D
Spectral cluster	NA	NA	NA

The spectral cluster algorithm was very temperamental and crashed at least once during the five repeats, due to an internal MATLAB error. Consequently, it was decided that it would not be an option in the deployed application because of its instability.

The density-based spatial clustering of applications with noise, Gaussian mixture algorithms and basic statistics correctly identified the damaged and undamaged blades, whilst classifying blade 1 as damaged, which was consistent with the results from the supervised models in Table 16.

5.5 Errors and Uncertainty

As with all projects that utilise real world data, there are potential sources of errors and uncertainties associated with the methods used, as well as the results. The exact magnitude of the errors was not calculated due to the Covid-19 restrictions, however the likely sources are discussed below.

For example, the error associated with the microphone itself, which could have been due to the out-of-date calibration or interference from other components. It was critical to ensure no magnetic fields, such as one from a solenoid, were close enough to the microphone to interfere with its operation.

An error could arise from the movement of the blade, owing to its non-rigid mounting and the small horizontal movement with the hammer shaft. These would both lead to a change in the impact location of the hammer, which would produce a slightly different response. In addition, as the actuator and components warm up, the hammer may not be dropped from the same location, or there may be less damping in the system, leading to a change in the impact energy.

The literature also alluded to the fact that the natural frequency would change depending on the temperature of the component, and in some cases could be as high as 10%, which may be more than the change due to the damage [74]. Therefore, without measuring the temperature and using it to standardise the frequencies, an error would be introduced.

5.6 Limitations of the Test Rig and Software Application

There are several limitations with the application and test rig. Firstly, the application extracts the highest peaks within ± 500 Hz of the estimated peak values. This value was recognised to perform with the three blades but may not be an appropriate value for other blades. In addition, the selected peaks of interest must be at least 1000 Hz apart from each other to prevent the incorrect peaks from being selected. If the peaks were poorly selected, this had a significant impact on the accuracy of the ML models.

Secondly, the locations of the natural frequency were the only features extracted. Nothing was extracted relating to the amplitude or width of the peak, nor to other factors which could vary between damaged and undamaged blades.

Thirdly, the impact hammer mechanism was only crudely constrained in the horizontal direction, which could produce less repeatable results.

6 Conclusion

This project has successfully completed its aim and corresponding objectives to develop a quick, reliable vibration testing method with ML algorithms to discern an undamaged AM turbine blade from a damaged one.

A review was initially completed of the relevant literature for the project. The testing method selected for the test rig was the non-destructive IET because of its ability to evaluate the entire component with a single impact and its ease of implementation with a non-contact microphone. The RUS method also showed promise, but due to the Covid-19 restrictions, it was decided to use the same method as the previous rig. It was found that with the IET, passing the raw signal through a FFT easily identified the natural frequency peaks, which could be used in the ML models. The literature identified several ML algorithms which could be used for supervised and unsupervised learning, which were subsequently compared.

The test rig was developed using the systematic engineering design process to ensure that it would be significantly better than the previous rig. The key result from the process was the test rig itself, which was in the process of being calibrated as the project ended. The creation of the GUI and sub-functions were developed using the software design process to make sure that the code was modular and could be developed further by another engineer. MATLAB was selected to create the code due to its vast number of built-in toolboxes, which would aid in the development of the application. The GUI itself was also a key result of the project, owing to its ability to perform all the operations and calculations required to classify a turbine blade, to collect training data for the ML model, to extract the key features and to create and train the various ML models.

Whilst only a limited amount of analysis was performed on the ML models due to the small dataset available, some key results were identified. The results may not be representative of an actual dataset and therefore more analysis would be required to confirm that the application still satisfied the aim. The results obtained from the supervised ML models highlighted that all conventional ML algorithms were much faster to train than the DL models and on average took 38 seconds, compared to 7 minutes for DL models, resulting in an 11 times reduction in training time. Furthermore, the accuracy of all models was 100% when classifying known blades. However, when classifying unknown blades, the DL models produced inconsistent results, supporting the literature that DL models are only accurate when trained using large datasets. In addition, the use of PCA was not advised with such a small dataset because of the limited amount of variation. The results from the unsupervised ML models highlighted that, apart from the spectral cluster algorithm, which was unreliable; the other algorithms all correctly predicted the condition of the blades.

The main theme of the analysis identified that the selection of natural frequency peaks was directly related to the accuracy of the ML models. This was expected, as the location of the peaks were the features used to train the ML models. For the dataset used, a small number of features were required to capture the variation between undamaged and damaged blades. Conversely, a large dataset would probably need more features and therefore the manual peak extraction process would become an even more important step.

In summary, with the limited dataset available, a test rig and GUI have been successfully created, capable of classifying the damage condition of previously unseen turbine blades using ML, a crucial factor when using AM components in aerospace applications.

7 Future Work

There are several additional objectives and features to be investigated which would improve the rig and the application. Many of these were unable to be completed due to the Covid-19 restrictions.

- The effect of temperature on the natural frequency values should be investigated to determine if the measurements need to be standardised.
- The effect of the width and amplitude of the natural frequency peaks should be investigated as to whether they are added to the extracted features used in the ML model.
- The different ML models should be reinvestigated, when a proper dataset is available, with more undamaged and damaged blades.
- A Gage R&R should be completed to determine the reproducibility and more importantly the repeatability of the rig, which may identify areas of improvement.
- The contact model could be completed with information regarding the stiffness and damping of the impact, which may help to design an improved impact mechanism.

8 References

[1] P. Kocovic, "History of Additive Manufacturing," ed: IGI Global, 2017, pp. 1-24.

[2] C. S. Tang and L. P. Veelenturf, "The strategic role of logistics in the industry 4.0 era," *Transportation research. Part E, Logistics and transportation review,* vol. 129, pp. 1-11, 2019, doi: 10.1016/j.tre.2019.06.004.

[3] U. M. Dilberoglu, B. Gharehpapagh, U. Yaman, and M. Dolen, "The Role of Additive Manufacturing in the Era of Industry 4.0," *Procedia manufacturing,* vol. 11, pp. 545-554, 2017, doi: 10.1016/j.promfg.2017.07.148.

[4] L. Langnau. "Laser sintering – What is it?" Make Parts Fast. https://www.makepartsfast.com/laser-sintering/ (accessed 25/04/21).

[5] Materialgeeza, "Selective laser melting system schematic," ed.

[6] S. Ford and M. Despeisse, "Additive manufacturing and sustainability: an exploratory study of the advantages and challenges," *Journal of cleaner production,* vol. 137, pp. 1573-1587, 2016, doi: 10.1016/j.jclepro.2016.04.150.

[7] 3DEO. "Environmental Impact of Additive Manufacturing." https://news.3deo.co/environmental-impact-of-additive-manufacturing (accessed 25/04/21).

[8] T. DebRoy *et al.*, "Additive manufacturing of metallic components – Process, structure and properties," *Progress in materials science,* vol. 92, no. C, pp. 112-224, 2018, doi: 10.1016/j.pmatsci.2017.10.001.

[9] E. Sacco and S. K. Moon, "Additive manufacturing for space: status and promises," *International journal of advanced manufacturing technology,* vol. 105, no. 10, pp. 4123-4146, 2019, doi: 10.1007/s00170-019-03786-z.

[10] M. J. "Additive manufacturing in aerospace is growing." 3D natives. https://www.3dnatives.com/en/additive-manufacturing-aerospace-growing-061220184/#! (accessed 25/04/21).

[11] M. Meo, "Final Year Project description and general discussion," N. Thomas, Ed., General details about the project and discussions ed, 2021

[12] YXLON-Aerospace. "The Right Solution for Your Aerospace Inspection Task." https://www.yxlon.com/en/applications/aerospace (accessed 25/04/21).

[13] D. Fumo. "Why Is Everyone Talking About Artificial Intelligence?" towardsdatascience. https://towardsdatascience.com/why-is-everyone-talking-about-ai-73bab31bf9c1 (accessed 25/04/21).

[14] N. Thomas, "Project Plan and Literature Review," University of Bath, 2020.

[15] J. Milewski, *Additive Manufacturing of Metals*, 1 ed. (Springer Series in Materials Science, no. 258). Springer International Publishing, 2017, p. 343.

[16] L. Schulenburg, "Inspection of Additive Manufactured (AM) parts using Computed Tomography (CT) ", 2018. [Online]. Available: https://visiconsult.de/inspection-of-additive-manufactured-am-parts-using-computed-tomography-ct/

[17] J. L. Bartlett, A. Jarama, J. Jones, and X. Li, "Prediction of microstructural defects in additive manufacturing from powder bed quality using digital image correlation," *Materials science & engineering. A, Structural materials : properties, microstructure and processing,* vol. 794, p. 140002, 2020, doi: 10.1016/j.msea.2020.140002.

[18] C. Mandache, "Overview of non-destructive evaluation techniques for metal-based additive manufacturing," *Materials science and technology,* vol. 35, no. 9, pp. 1007-1015, 2019, doi: 10.1080/02670836.2019.1596370.

[19] S. Tammas-Williams, P. J. Withers, I. Todd, and P. B. Prangnell, "The Effectiveness of Hot Isostatic Pressing for Closing Porosity in Titanium Parts Manufactured by Selective Electron Beam Melting," *Metallurgical and Materials Transactions A,* vol. 47, no. 5, pp. 1939-1946, 2016/05/01 2016, doi: 10.1007/s11661-016-3429-3.

[20] J.-P. a. S. G.-H. a. Y. S. a. Y. D.-Y. a. L. J.-S. a. B. M. a. Y. J.-H. Choi, "Densification and microstructural investigation of Inconel 718 parts fabricated by selective laser melting," *Powder Technology,* vol. 310, 2017, doi: 10.1016/j.powtec.2017.01.030.

[21] J. J. Lewandowski and M. Seifi, "Metal Additive Manufacturing: A Review of Mechanical Properties," *Annual Review of Materials Research,* vol. 46, no. 1, pp. 151-186, 2016, doi: 10.1146/annurev-matsci-070115-032024.

[22] Vibrant, "Process Compensated Resonance Testing = PCRT." [Online]. Available: https://www.vibrantndt.com/about-pcrt/

[23] A. Lim, "What is Natural Frequency." [Online]. Available: https://www.thoughtco.com/natural-frequency-4570958

[24] B. S. P. British Standards (BS EN ISO 17296-3:2016), "BS EN ISO 17296-3:2016: Additive manufacturing. General principles. Main characteristics and corresponding test methods," ed: British Standards Institute, 2014.

[25] H. Hobart Institute of Welding Technology, *Destructive testing methods training workbook*. Place of publication not identified: Place of publication not identified Hobart Institute of Welding Technology, 2010.

[26] P. Charalampous, I. Kostavelis, and D. Tzovaras, "Non-destructive quality control methods in additive manufacturing: a survey," *Rapid prototyping journal,* vol. 26, no. 4, pp. 777-790, 2020, doi: 10.1108/RPJ-08-2019-0224.

[27] B. Kamsu-Foguem, "Knowledge-based support in Non-Destructive Testing for health monitoring of aircraft structures," *Advanced engineering informatics,* vol. 26, no. 4, pp. 859-869, 2012, doi: 10.1016/j.aei.2012.06.006.

[28] L. Cartz, *Nondestructive testing : radiography, ultrasonics, liquid penetrant, magnetic particle, eddy current*. Materials Park, OH: Materials Park, OH : ASM International, 1995.

[29] F. Ali Sophian Guiyun Tian Mengbao, "Pulsed Eddy Current Non-destructive Testing and Evaluation ： A Review," *Chinese journal of mechanical engineering,* vol. 30, no. 3, pp. 500-514, 2017, doi: 10.1007/s10033-017-0122-4.

[30] S. Carmignato, W. Dewulf, and R. Leach, *Industrial X-Ray Computed Tomography*, 1st ed. 2018. ed. Cham: Cham : Springer International Publishing : Imprint: Springer, 2018.

[31] T. J. Society for Non-Destructive Inspection, *Practical Acoustic Emission Testing*, 1st ed. 2016. ed. Tokyo: Tokyo : Springer Japan, 2016.

[32] C. U. Grosse and M. Ohtsu, *Acoustic Emission Testing*, 1st ed. 2008. ed. (With contributions by numerous experts). Berlin, Heidelberg: Berlin, Heidelberg : Springer Berlin Heidelberg : Imprint: Springer, 2008.

[33] F. Tian *et al.*, "An Ultrasonic Pulse-Echo Method to Detect Internal Defects in Epoxy Composite Insulation," *Energies (Basel),* vol. 12, no. 24, p. 4804, 2019, doi: 10.3390/en12244804.

[34] Z. Zou *et al.*, "An Ultrasonic Longitudinal Through-Transmission Method to Measure the Compressive Internal Stress in Epoxy Composite Specimens of Gas-Insulated Metal-Enclosed Switchgear," *Energies (Basel),* vol. 13, no. 5, p. 1248, 2020, doi: 10.3390/en13051248.

[35] M. Hardy *et al.*, "33.6.2 Extension to Complex Geometries," in *Superalloys 2016 - Proceedings of the 13th International Symposium on Superalloys*: TMS (The Minerals, Metals & Materials Society), pp. 303-312.

[36] F. F. Balakirev, S. M. Ennaceur, R. J. Migliori, B. Maiorov, and A. Migliori, "Resonant ultrasound spectroscopy: The essential toolbox," *Review of scientific instruments,* vol. 90, no. 12, pp. 121401-121401, 2019, doi: 10.1063/1.5123165.

[37] I. Solodov, A. Dillenz, and M. Kreutzbruck, "A new mode of acoustic NDT via resonant air-coupled emission," *Journal of applied physics.,* vol. 121, no. 24, p. 245101, 2017, doi: 10.1063/1.4985286.

[38] Vibrant, "PCRT Resonance Solutions," in "White Paper," Vibrant, 2018. [Online]. Available: https://www.vibrantndt.com/advanced-technology/

[39] L. Hunter, "Reliable Modern NDT," Albuquerque, USA. [Online]. Available: https://www.ndt.net/article/reliability2009/Inhalt/fr2a3.pdf

[40] J. Rossin *et al.*, "Assessment of grain structure evolution with resonant ultrasound spectroscopy in additively manufactured nickel alloys," *Materials characterization,* vol. 167, p. 110501, 2020, doi: 10.1016/j.matchar.2020.110501.

[41] B. Psiuk, P. Wiecinska, B. Lipowska, E. Pietrzak, and J. Podwórny, "Impulse Excitation Technique IET as a non-destructive method for determining changes during the gelcasting process," *Ceramics international,* vol. 42, no. 3, pp. 3989-3996, 2016, doi: 10.1016/j.ceramint.2015.11.067.

[42] G. Roebben, B. Bollen, A. Brebels, J. Van Humbeeck, and O. Van der Biest, "Impulse excitation apparatus to measure resonant frequencies, elastic moduli, and internal friction at room and high temperature," *Review of scientific instruments,* vol. 68, no. 12, pp. 4511-4515, 1997, doi: 10.1063/1.1148422.

[43] J. Berrio, "Characterization of effective Young's modulus for Fused Deposition Modeling manufactured topology optimization designs," *International Journal of Advanced Manufacturing Technology,* vol. 103, 2019, doi: 10.1007/s00170-019-03747-6.

[44] M. D. Fariñas, D. Jimenez-Carretero, D. Sancho-Knapik, J. J. Peguero-Pina, E. Gil-Pelegrín, and T. Gómez Álvarez-Arenas, "Instantaneous and non-destructive relative water content estimation from deep learning applied to resonant ultrasonic spectra of plant leaves," *Plant methods,* vol. 15, no. 1, pp. 128-128, 2019, doi: 10.1186/s13007-019-0511-z.

[45] E. Touger. "What's the Difference Between Artificial Intelligence (AI), Machine Learning, and Deep Learning?" Prowess. https://www.prowesscorp.com/whats-the-difference-between-artificial-intelligence-ai-machine-learning-and-deep-learning/ (accessed 25/04/21).

[46] D. Fumo. "Types of Machine Learning Algorithms You Should Know." towards data science. https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861 (accessed 25/11/2020).

[47] J. Brownlee. "14 Different Types of Learning in Machine Learning." Machine Learning Mastery. https://machinelearningmastery.com/types-of-learning-in-machine-learning/ (accessed 25/11/2020).

[48] MathWorks, *Mastering Machine Leaning A Step-by-Step Guide with MATLAB*, 2019. [Online]. Available: https://www.mathworks.com/campaigns/offers/mastering-machine-learning-with-matlab.html.

[49] MathWorks. "What is Machine Learning?" MathWorks. https://www.mathworks.com/discovery/machine-learning.html (accessed 25/04/21).

[50] S. Ghosh *et al.*, "One-component order parameter in URu 2 Si 2 uncovered by resonant ultrasound spectroscopy and machine learning," *Science advances.,* vol. 6, no. 10, p. eaaz4074, 2020, doi: 10.1126/sciadv.aaz4074.

[51] A. Paral, D. K. Singha Roy, and A. K. Samanta, "A deep learning-based approach for condition assessment of semi-rigid joint of steel frame," *Journal of Building Engineering,* vol. 34, p. 101946, 2021, doi: 10.1016/j.jobe.2020.101946.

[52] M. Innat. "Preferable tools for machine learning - Python - MatLab - R." codementor community. https://www.codementor.io/@innat_2k14/preferable-tools-for-machine-learning-python-matlab-r-jfozzpphz (accessed 25/11/2020).

[53] C. D. Costa. "Best Python Libraries for Machine Learning and Deep Learning." towards data science. https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c (accessed 25/11/2020).

[54] C. Nnamdi. "Top 5 Machine Learning Libraries." Bits and Pieces. https://blog.bitsrc.io/top-5-javascript-machine-learning-libraries-604e52acb548 (accessed 25/11/2020).

[55] M. Miskuf, P. Michalik, and I. Zolotova, "Data mining in cloud usage data with Matlab's statistics and machine learning toolbox," ed: IEEE, 2017, pp. 000377-000382.

[56] engineeringtoolbox. "Sound Absorption Coefficient." The Engineering Toolbox. https://www.engineeringtoolbox.com/accoustic-sound-absorption-d_68.html (accessed 25/04/21).

[57] *Datasheet MSS09/031 AGFT Whis UV*. [Online]. Available: https://uk.rs-online.com/web/p/acoustic-insulation/1034068/. Accessed: (25/04/21).

[58] primacoustic. "Science of Absorption " https://www.primacoustic.com/broadway-panels/science/ (accessed 25/04/21).

[59] N. Bailey, "Mechanical Vibrations and Noise - Noise (ME30033)," University of Bath, 2020.

[60] thePhysicsClassroom. "The Speed of Sound " https://www.physicsclassroom.com/class/sound/Lesson-2/The-Speed-of-Sound (accessed 25/04/21).

[61] H. Sohn *et al.*, "A Review of Structural Health Monitoring Literature: 1996–2001," Los Alamos National Laboratory, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=43453048A5725D068F53552AE67579DB?doi=10.1.1.729.3993&rep=rep1&type=pdf

[62] W. Nick, J. Shelton, K. Asamene, and A. Esterline, "A Study of Supervised Machine Learning Techniques for Structural Health Monitoring," in *MAICS*, 2015. [Online]. Available: https://www.semanticscholar.org/paper/A-Study-of-Supervised-Machine-Learning-Techniques-Nick-Shelton/cd776ebbc8cab9a36554e9b03940e2b6b94ab5ff. [Online]. Available: https://www.semanticscholar.org/paper/A-Study-of-Supervised-Machine-

Learning-Techniques-Nick-Shelton/cd776ebbc8cab9a36554e9b03940e2b6b94ab5ff

[63] S. W. Doebling, C. R. Farrar, M. B. Prime, and D. W. Shevitz, "Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: A literature review," Los Alamos National Laboratory, 1996. [Online]. Available: https://www.osti.gov/biblio/249299-damage-identification-health-monitoring-structural-mechanical-systems-from-changes-vibration-characteristics-literature-review

[64] MathWorks. "What is Deep Learning?" MathWorks. https://www.mathworks.com/discovery/deep-learning.html (accessed 25/04/21).

[65] MathWorks, *Machine learning vs deep learning*: MathWorks. [Online]. Available: https://explore.mathworks.com/machine-learning-vs-deep-learning.

[66] J. Morrill and B. Smus. "SPECTROGRAM." https://musiclab.chromeexperiments.com/spectrogram (accessed 25/04/21).

[67] MathWorks. "Pretrained Deep Neural Networks." https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html (accessed 25/04/21).

[68] MathWorks. "Unsupervised Learning." MathWorks. https://www.mathworks.com/discovery/unsupervised-learning.html (accessed 25/04/21).

[69] J. Skála, "Algorithms for manipulating large geometric data," Faculty of applied sciences, University of West Bohemia, 2012. [Online]. Available: https://www.researchgate.net/publication/258704984_Algorithms_for_manipulating_large_geometric_data

[70] MathWorks. "Bayesian Optimization Algorithm." https://www.mathworks.com/help/stats/bayesian-optimization-algorithm.html (accessed 25/04/21).

[71] Farnell. "PICOSCOPE 4424 KIT." https://uk.farnell.com/pico-technology/picoscope-4424-kit/oscilloscope-pc-high-precision/dp/1669213?ost=1669213 (accessed 25/04/21).

[72] *Ultrasonic Transducers*. [Online]. Available: http://ultrangroup.com/wp-content/uploads/transducer-catalog.pdf.

[73] Quality-One. "Gage Repeatability & Reproducibility." https://quality-one.com/grr/ (accessed 25/04/21).

[74] L. Zhao and H. Jin, "Effect of temperature on natural frequencies of bridge structure of metallurgical crane," *IOP conference series. Materials Science and Engineering,* vol. 1043, no. 4, p. 42047, 2021, doi: 10.1088/1757-899X/1043/4/042047.

9 Appendix

Appendix A – RunSingleTest Function Description and Code

RunSingleTest.m was created to call all the sub-functions required to conduct a single test outputting the class prediction and probability.

```
function [dataOut,classPred,probability,status] = ...
    RunSingleTest(MLmodel,testType,filePath)
% This function is called from the Pass_Fail application
% It contains the logic required to perform a single test by calling all
% of the helper functions

% Function inputs:
% MLmodel - Structure containing the machine learning model to be used
% testType - If the test is simulated or using hardware
% filePath - The file path of the simulated file

% Function outputs:
% dataOut - Structure containing all the outputs from the helper functions
% classPred - The predicted classification ('UD' or 'D')
% probability - The probability the blade is 'UD'
% status - The result of the function (any errors)
% =========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% -------------------------------------------------------------------------

% Function begins:
% Construct status struct which is passed through functions and is
% reported back to the UI
status.error = 0;
status.desc = "";
dataOut = struct();
classPred = "NA";
probability = "NA";

try
% Import Data and process it
    if testType == "Live"
        [data,status] = ImpactTest("No",0,"","");
        if status.error == 1
            % There has been a problem with the oscilloscope
            return
        end

    elseif testType == "Simulated"
        % Import CSV file and run simulated test
        data = readtable(filePath);
    else
        status.error = 1;
        status.desc = "Incorrect Test Type";
    end

    % Call the PreprocessData function
    preprocessedData = PreprocessData(data);

    % Call the ExtractFeature function
    ExtractedFeaturesData = ExtractFeature(preprocessedData,...
        MLmodel.usedFreqPeaks);

    dataOut = ExtractedFeaturesData;
```

```
    if MLmodel.MLmodelType == "DL"
        Im = CreateSpectrogram("No","Test",filePath,data);
        % Resize the image for 224x224 as that is the model input
        dataTest = imresize(Im,[224 224]);
    else
        % Remove the unused columns
        dataTest = removevars(struct2table(dataOut,'AsArray',true),...
            {'rawSig','rawSigTime','f','amp','phase','extractedFreqs',...
            'ampAtExtractedFreqs'});
    end

    % Apply ML
    [classPred,probability] = Classify(MLmodel,dataTest);

    status.error = 0;
    status.desc = "";
catch ME
    status.error = 1;
    status.desc = append('RunSingleTest.m: ',ME.message);
end

end
```

Appendix B – ImpactTest Function Description and Code

ImpactTest.m was written to call all the sub-functions required to produce an impact and collect the acoustic data from the microphone.

```
function [dataOut,status] = ImpactTest(saveCSV,testIdx,folderPath,...
    testFileName)
% This function is called from the Pass_Fail application or RunSingleTest
% It contains the logic required to perform an impact, record the impact,
% save the impact to CSV by calling all of the helper functions

% Function inputs:
% saveCSV - Yes or No for if the csv file should be saved
% testIdx - The index that should be appended to the test file name
% folderPath - If the test is simulated or using hardware
% testFileName - The file name of the blade

% Function outputs:
% dataOut - Structure containing all the outputs from the helper functions
% status - The result of the function (any errors)
% =========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% -------------------------------------------------------------------------

% Call functions to conduct the impact and record the data
[ps4000DeviceObj,blockGroupObj,timeIntervalNanoseconds,status] = ...
    ConfigurePS4000();
if status.error == 1
    dataOut = struct();
    return
end

try
    hammerDropJob = batch(@DropHammer,1); % Parallel job as reading is blocking
    tStart = datetime;
    invoke(blockGroupObj,'runBlock',0); % Blocking

    [numSamples,~,chA] = invoke(blockGroupObj,'getBlockData',0,0,1,0);
    timeNs = double(timeIntervalNanoseconds) * double(0:numSamples - 1);
    timeMs = (timeNs / 1e6)';
    data = table(timeMs,chA,'VariableNames',{'Time','ChannelA'});

    % res = fetchOutputs(hammerDropJob);

    tEnd = datetime;
    if seconds(diff(datetime([tStart;tEnd]))) > 15
        % The oscilloscope was not trigger, no impact
        status.error = 1;
        status.desc = ['The oscilloscope was not triggered, ',...
            'the threshold may be set too high or there was not an impact'];
        throw(MException('MATLAB:noTrigger','Scope not triggered'));
    end

    % Save file to CSV
    if saveCSV == "Yes"
        filename = fullfile(folderPath,testFileName + "-" + testIdx + ".csv");
        writetable(data,filename)
    end

    dataOut = data;
    status.error = 0;
    status.desc = "";
catch ME
    dataOut = struct();
```

```
end

invoke(ps4000DeviceObj,'ps4000Stop');
disconnect(ps4000DeviceObj);
delete(ps4000DeviceObj);

end
```

Appendix C – ConfigurePS4000 Function Description and Code

ConfigurePS4000.m was created to initialise the PicoScope 4424 oscilloscope by connecting to it and setting parameters such as, the sampling frequency, the trigger threshold voltage and the quantity of data to record.

```
function [ps4000DeviceObj,blockGroupObj,timeIntervalNanoseconds,status] = ...
    ConfigurePS4000()
% This function configures the PS4000 scope ready to record data
% https://www.mathworks.com/matlabcentral/fileexchange/
% 49117-picoscope-4000-series-matlab-generic-instrument-driver
% PS4000_ID_Block_Example as an example

% Function inputs:
% Change timeBaseIndex to change the sampling rate
% Change threshold to change the minimum trigger voltage

% Function outputs:
% ps4000DeviceObj - The oscilloscope object handle
% blockGroupObj - The block data extraction process
% timeIntervalNanoseconds - The sample time step
% status - The result of the function (any errors)

% ========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% ------------------------------------------------------------------------

PS4000Config; % Obtain internal values and files required for scope

% Device connection
if (exist('ps4000DeviceObj', 'var') && ...
        ps4000DeviceObj.isvalid && strcmp(ps4000DeviceObj.status, 'open'))

    % Close connection to device
    disconnect(ps4000DeviceObj);
    delete(ps4000DeviceObj);

else
    ps4000DeviceObj = struct();
    blockGroupObj = struct();
    timeIntervalNanoseconds = 0;

    % Create a device object. Note the device must be plugged in
    try
        % Try to create an icdevice from the driver. This often fails
        ps4000DeviceObj = icdevice('picotech_ps4000_generic.mdd');
    catch ME
        disp(ME.message);
        status.error = 1;
        status.desc = ['Problem connecting to the oscilloscope, ',...
            'check it is connected correctly and if so power cycle'];
        return
    end

    try
        % Try to connect device object to hardware. This often fails
        connect(ps4000DeviceObj);
    catch ME
        disconnect(ps4000DeviceObj);
        delete(ps4000DeviceObj);
        disp(ME.message);
        status.error = 1;
        status.desc = ['Problem connecting to the oscilloscope, ',...
            'try another USB hole, if all have been tried, restart the computer'];
```

```
        return
    end

    try
        % Set Channels to use
        % Channels       : 1 - 3 (PS4000_CHANNEL_B - PS4000_CHANNEL_D)
        % Enabled        : 0
        % Type           : 1 (DC)
        % Range          : 8 (ps4000Enuminfo.enPS4000Range.PS4000_5V)

        % Turn channel B off as default and only use A
        [Status.setChB] = invoke(ps4000DeviceObj,'ps4000SetChannel',1,0,1,8);

        if (ps4000DeviceObj.channelCount == PicoConstants.QUAD_SCOPE)

            % Turn channel C and D off if 4 channel scope
            [Status.setChC] =
invoke(ps4000DeviceObj,'ps4000SetChannel',2,0,1,8,0.0,0);
            [Status.setChD] =
invoke(ps4000DeviceObj,'ps4000SetChannel',3,0,1,8,0.0,0);

        end

        % Specify the sampling rate and maximum number of samples
        % Sampling time step = (timeBaseIndex - 1) / 20,000,000
        % Sampling time step = (5 - 1) / 20,000,000 = 200 ns

        Status.getTimebase2 = PicoStatus.PICO_INVALID_TIMEBASE;
        timeBaseIndex = 5; % 200 ns

        while (Status.getTimebase2 == PicoStatus.PICO_INVALID_TIMEBASE)

            [Status.getTimebase2,timeIntervalNanoseconds,maxSamples] = ...
                invoke(ps4000DeviceObj,'ps4000GetTimebase2',timeBaseIndex, 0);

            if (Status.getTimebase2 == PicoStatus.PICO_OK)
                break;
            else
                timeBaseIndex = timeBaseIndex + 1;
            end

        end

        fprintf('Timebase index: %d, sampling interval: %.1f ns\n',...
            timeBaseIndex,timeIntervalNanoseconds);

        % Configure the device |timebase| property value.
        set(ps4000DeviceObj,'timebase',timeBaseIndex);

        % Set up the trigger for the impact
        triggerGroupObj = get(ps4000DeviceObj,'Trigger');
        triggerGroupObj = triggerGroupObj(1);

        [Status.setTriggerOff] = invoke(triggerGroupObj,'setTriggerOff');

        % Set device to trigger automatically after 15 seconds
        % There must be a problem if there is no impact
        set(triggerGroupObj,'autoTriggerMs',15000);
    %
    %     % Channel     : 0 (ps4000Enuminfo.enPS4000Channel.PS4000_CHANNEL_A)
    %     % Threshold   : 500 (mV)
    %     % Direction   : 2
(ps4000Enuminfo.enPS4000ThresholdDirection.PS4000_RISING)
        threshold = 10;
        [Status.SimpleTrigger] = invoke(triggerGroupObj,...
            'setSimpleTrigger',0,threshold,2);
```

```
        % Set up the block parameters
        blockGroupObj = get(ps4000DeviceObj,'Block');
        blockGroupObj = blockGroupObj(1);

        % Set pre-trigger and post-trigger samples as required - the total of
        % this should not exceed the value of maxSamples

        numPointsBefore = round(10e-3 / timeIntervalNanoseconds/1e-9); % 10ms
before
        numPointsAfter = round(190e-3 / timeIntervalNanoseconds/1e-9); % 190ms
after
        set(ps4000DeviceObj,'numPreTriggerSamples',numPointsBefore);
        set(ps4000DeviceObj,'numPostTriggerSamples',numPointsAfter);

        %[Status.runBlock] = invoke(blockGroupObj,'runBlock',0);
        %[numSamples,overflow,chA] = invoke(blockGroupObj,'getBlockData',0,0,1,0);

        %[Status.runBlock] = invoke(ps4000DeviceObj,'ps4000RunBlock', 0);
        %ps4000IsReady
        %ps4000GetValues

    %     timeNs = double(timeIntervalNanoseconds) * double(0:numSamples - 1);
    %     timeMs = (timeNs / 1e6)';
    %     data = table(timeMs,chA,'VariableNames',{'Time','ChannelA'});
    %     plot(timeMs,chA);
    status.error = 0;
    status.desc = '';

    catch ME
        % Disconnect device object from hardware.
        invoke(ps4000DeviceObj,'ps4000Stop');
        disconnect(ps4000DeviceObj);
        delete(ps4000DeviceObj);
    end
end

end
```

Appendix D – DropHammer Function Description and Code

DropHammer.m was written to tell the Arduino to drop the hammer and to therefore create an impact on the test blade. It is incomplete due to the electronic technicians not completing the firmware of the Arduino as the report was submitted.

```
function [status] = DropHammer()
% This function communicates with the Arduino to drop the hammer

% Function inputs:

% Function outputs:
% status - The result of the function (any errors)

% ========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% ------------------------------------------------------------------------

%https://uk.mathworks.com/matlabcentral/answers/
%325725-sending-values-from-matlab-to-arduino-using-serial-communication

%a = arduino('Uno');
%[dataOut,payloadSize] = sendCommand(arduinoObj,libName,commandID,dataIn,timeout)
end
```

Appendix E – PreprocessData Function Description and Code

PreprocessData.m was created to pre-process the raw signal data from the oscilloscope and microphone. The input of the function is the time series voltage signal for a single test, as seen in Figure 8. The function filters the signal to remove very low frequencies and resets the time to zero at the first datapoint. It then performs an FFT on the signal, outputting the amplitude and phase values in the frequency domain.

```
function dataOut = PreprocessData(data)
%% This function preprocesses the input data
% 1 - cleans the data with low and high pass filters
% 2 - performs an FFT to extract the amplitude and phase against frequency

% Function inputs:
% data - The timeseries data from the sensor

% Function outputs:
% dataOut - Structure containing the result of the FFT
% ========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% ------------------------------------------------------------------------

% Process the data
dt = (data.Time(2) - data.Time(1))/1000;
fs = 1/dt;

% Remove low frequencies below 100 Hz
data.ChannelA = highpass(data.ChannelA,100,fs);
% Reset the time to 0 for the first data point
if data.Time(1) < 0
    data.Time = data.Time + abs(data.Time(1));
else
    data.Time = data.Time - abs(data.Time(1));
end

dataOut.rawSig = data.ChannelA;
dataOut.rawSigTime = data.Time;

% Perform the FFT
y = data.ChannelA;
NFFT = length(y);
Y = fft(y)/NFFT; % N-point complex DFT

dataOut.f = fs/NFFT*(0:(NFFT/2))';

% Calculate the amplitude
P2 = abs(Y);
P1 = P2(1:NFFT/2+1);
P1(2:end-1) = 2*P1(2:end-1);

dataOut.amp = P1;

% Calculate the phase
Ymodified = Y;
threshold = max(P1)*(1/100); % 1% above
Ymodified(abs(Y) < threshold) = 0;

phase2 = atan2(imag(Ymodified),real(Ymodified));
phase1 = phase2(1:NFFT/2+1);

dataOut.phase = phase1;
end
```

Appendix F – ExtractFeautre Function Description and Code

ExtractFeautre.m was written to identify the key features of the amplitude in the frequency domain from the output of the pre-processing function. The input is the pre-processed data and the estimated location of the natural frequencies identified by the user on the retrain machine learning algorithm tab of the GUI. The output is the exact natural frequency value of the peaks within a tolerance of the estimated natural frequency values.

```
function dataOut = ExtractFeature(data,freqPeaks)
% This function extracts features from the input signal data

% Function inputs:
% data - data from the preprocessed stage
% freqPeaks - Array of estimated locations of the natural frequency peaks

% Function outputs:
% dataOut - Structure containing the result of preprocessed stage and the
% extracted features
% ========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% ------------------------------------------------------------------------

% Extract Features below
dataOut = data;

% Identify peaks and therefore natural frequency modes
% More feature selection can be implemented here

if ~isempty(freqPeaks)
    for i = 1:numel(freqPeaks)
        % Cutoff frequencies may need to change from 500 Hz in the future!
        midFreq = freqPeaks(i) * 1000; % Convert to Hz
        botFreq = midFreq - 500; % Bottom cutoff
        [~,botFreqIdx] = min(abs(data.f - botFreq)); % Bottom cutoff index
        topFreq = midFreq + 500; % Top cutoff
        [~,topFreqIdx] = min(abs(data.f - topFreq)); % Top cutoff index
        [~,maxIdx] = max(data.amp(botFreqIdx:topFreqIdx)); % Index of peak

        % Need -1 as both include the bottom frequency index
        maxFreqIdx = botFreqIdx + maxIdx - 1;
        extractedFreq(i) = data.f(maxFreqIdx);
        ampAtExtractedFreq(i) = data.amp(maxFreqIdx);

        name = ['Feat',num2str(i)];
        dataOut.(name) = extractedFreq(i);
    end

    % All the extracted fequencies in an array for ease of plotting
    dataOut.extractedFreqs = extractedFreq;
    dataOut.ampAtExtractedFreqs = ampAtExtractedFreq;
end

end
```

Appendix G – CreateSpectrogram Function Description and Code

CreateSpectrogram.m was created to use a saved CSV file or data from a live test to create spectrograms, which could be saved to either use in the training of a DL model or to be classified. The outputted image was processed to remove all axis information and the outside border, to leave only the spectrum, see Figure 33.

```
function im = CreateSpectrogram(saveCSV,type,filePath,data)
% This function parses the CSV data file and creates a spectrogram
% The spectrogram is saved in the same location as the CSV file

% Function inputs:
% saveCSV - Yes or No if the spectrogram should be saved
% type - Training or Test if the data is read from a file or passed in
% filePath - The file path of the CSV file
% data - If a 'Test' then the data is passed in, if not it is read

% Function outputs:
% im - Matrix array of RGB information for the image

% ========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% ------------------------------------------------------------------------

% Extract image data from spectrogram
if type == "Training"
    % Load the data from the CSV file
    [path,file,~] = fileparts(filePath);
    data = readtable(filePath);
else
    path = pwd;
    file = 'temp';
end

% Preprocess the data
dt = (data.Time(2) - data.Time(1))/1000;
fs = 1/dt;

% Remove low frequencies below 100 Hz
data.ChannelA = highpass(data.ChannelA,100,fs);

% Standardise the data 2ms before the impact and have 173ms of data after
try
    noiseThreshold = max(data.ChannelA)*(10/100); % Noise 10% of max measurement
    impactIdxs = find(data.ChannelA >= noiseThreshold);
    impactIdx = impactIdxs(1) - round(0.002/dt); % Keep data 2ms before impact
    data = data(impactIdx:end,:);

    endIdx = round(0.175/dt); % Keep 175ms of data
    data = data(1:endIdx,:);
catch
    % Data is above 1 or not 175ms of data
end

if data.Time(1) < 0
    data.Time = data.Time + abs(data.Time(1));
end

pspectrum(data.ChannelA,fs,"spectrogram");
ylim([0 100]) % Limit to 100 kHz
axis off;
colorbar off;
title("");
```

```
newFilePath = fullfile(path,append(file,'.png'));
exportgraphics(gcf,newFilePath,'BackgroundColor','none','Resolution',100);
close

im = imread(newFilePath);

if saveCSV == "Yes"
    % File is already saved
else
    delete(newFilePath);
end

end
```

Appendix H – CreateTrainMLAlgorithm Function Description and Code

CreateTrainMLAlgorithm.m was written to use the training data to create and then train the algorithm which outputs a trained model, ready to be used to classify the blades. The function creates a model as specified by the input and can automatically conduct PCA whilst optimising the hyperparameters of the ML algorithm. Furthermore, the layers that needed to be changed of a pre-trained DL model were automatically replaced.

```
function [mdlAcc,mdlLoss,classTrue,classPred,usedFreqPeaks,status] = ...
    CreateTrainMLAlgorithm(MLmodelType,trainingDataUsed,usePCA,...
    freqPeaks,trainingDataFolderPath,applicationPath)
% This function is called from the Pass_Fail application
% It contains the code to train all of the different machine learning algorithms

% Function inputs:
% MLmodelType - The machine learning model method ('DL','kNN',ect)
% trainingDataUsed - The type of training data available (sup or unsup)
% usePCA - Use principal component analysis ('Yes' or 'No')
% freqPeaks - The estimated frequency peaks
% trainingDataFolderPath - The path to where the training data is kept
% applicationPath - The path to where the Pass_Fail app is stored

% Function outputs:
% mdlAcc - The accuracy of the trained model
% mdlLOss - The loss of the trained model
% classTrue - The actual class of the test data
% classPred - The predicted class of the test data
% usedFreqPeaks - The estimated frequency peaks which were used to train
% the model
% status - The result of the function (any errors)
% ========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% ------------------------------------------------------------------------

% Construct status struct
% This will be pass around the application and reported back to the UI
status.error = 0;
status.desc = "";
mdlAcc = NaN;
mdlLoss = NaN;
classTrue = {};
classPred = {};
usedFreqPeaks = freqPeaks;
features = length(usedFreqPeaks);


% Create datastores and extract features
try
    if MLmodelType == "DL"
        % Need an image datastore
        files = dir(fullfile(trainingDataFolderPath,'**/*.csv'));
        for i = 1:length(files)
            name = files(i).name;
            folder = files(i).folder;
            CreateSpectrogram("Yes","Training",fullfile(folder,name),'');
        end

        % Create the damaged image datastore
        damagedBladeFolderPath = fullfile(trainingDataFolderPath,'Damaged');
        im_Damaged_ds = imageDatastore(damagedBladeFolderPath,...
            'IncludeSubfolders',true);
        im_Damaged_ds.Labels = repelem("D",numel(im_Damaged_ds.Files));
```

```
        % Create the undamaged image datastore
        undamagedBladeFolderPath = fullfile(trainingDataFolderPath,'Undamaged');
        im_Undamaged_ds = imageDatastore(undamagedBladeFolderPath,...
            'IncludeSubfolders',true);
        im_Undamaged_ds.Labels = repelem("UD",numel(im_Undamaged_ds.Files));

        % Create the combined image datastore with corresponding labels
        im_ds = imageDatastore([im_Damaged_ds.Files; im_Undamaged_ds.Files]);
        im_ds.Labels = [categorical(im_Damaged_ds.Labels);...
            categorical(im_Undamaged_ds.Labels)];

    else
        % Need a tabular datastore
        if trainingDataUsed == "Undamaged"
            % Only the undamaged folder
            undamagedFolderPath = fullfile(trainingDataFolderPath,'Undamaged');
            files_ds = tabularTextDatastore(undamagedFolderPath,...
                'NumHeaderLines',2,'VariableNames',["Time","ChannelA"],...
                'IncludeSubfolders',true,'FileExtensions',{'.csv'},...
                'ReadSize','file');
        else
            % Both folders required
            files_ds = tabularTextDatastore(trainingDataFolderPath,...
                'NumHeaderLines',2,'VariableNames',["Time","ChannelA"],...
                'IncludeSubfolders',true,'FileExtensions',{'.csv'},...
                'ReadSize','file');

            % Extract the class of the blades
            [~,files,~] = fileparts(files_ds.Files);
            classes = extractBetween(string(files),'_','-');
            knownClass = categorical(classes);
        end

        % Call the PreprocessData function for the datastore
        preprocessedFiles_ds = transform(files_ds,...
            @(data) PreprocessData(data));

        % Call the ExtractFeature function for the datastore
        ExtractedFeatures_ds = transform(preprocessedFiles_ds,...
            @(data) ExtractFeature(data,freqPeaks));

        % Extract the data and remove the unused columns
        dataOut = struct2table(readall(ExtractedFeatures_ds));
        data = removevars(dataOut,{'rawSig','rawSigTime','f','amp',...
            'phase','extractedFreqs','ampAtExtractedFreqs'});

        if usePCA == "Yes"
            % Perform PCA to reduce less useful features and standarise
            % https://www.mathworks.com/help/stats/pca.html
            [pcCoeff,transformedData,~,~,percentexp,mean] = ...
                pca(table2array(data));

            totalPercentage = 0;
            for idx = 1:length(percentexp)
                % Output index where it has 97.5% variation
                totalPercentage = totalPercentage + percentexp(idx);
                if totalPercentage >= 97.5
                    break
                end
            end

            features = idx;
            featReduction = '-PCA-';
            PCA = struct('coeff',pcCoeff,'mu',mean,'index',idx);
            dataMat = transformedData(:,1:idx);
            data = array2table(dataMat);
        else
            featReduction = '-';
```

```
            PCA = struct();
        end
    end

catch ME
    status.error = 1;
    status.desc = append('Error in CreateTrainMLAlgorithm.m: ',ME.message);
    return
end


% Create the machine learning models
% This is where new models can be added in the switch case structure
try
    if trainingDataUsed == "Undamaged"
        % Train undamaged models
        mdl = data;
        trainingData = "UD only";
        mdlAcc = "NA";
        mdlLoss = "NA";

        switch MLmodelType
            case 'Basic'
                fileName = ['model-Basic',featReduction,'Features-',...
                    num2str(features)];
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','usedFreqPeaks','trainingData','mdlAcc');

            case 'DBSCAN'
                % Fit the correct epsilon for the undamaged training data
                minpts = features + 1;
                epsilonTest = 1:1:100;
                for epsilon = epsilonTest
                    classes = dbscan(table2array(data),epsilon,minpts);
                    outlier(epsilon) = ismember(-1,classes);
                end

                nonOutlierIdx = find(outlier == 0);
                firstIdx = nonOutlierIdx(1);
                epsilon = epsilonTest(firstIdx); % Required epsilon

                fileName = ['model-DBSCAN',featReduction,'Features-',...
                    num2str(features)];
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','usedFreqPeaks','trainingData','mdlAcc',...
                    'minpts','epsilon');

            case 'Gaussian'
                % Fit distributions to find values of AIC and BIC which
                % show the quality of fit for 1-10 clusters
                for k = 1:10
                    GMModels{k} = fitgmdist(table2array(data),k,...
                        'RegularizationValue',0.01);
                    AIC(k) = GMModels{k}.AIC;
                    BIC(k) = GMModels{k}.BIC;
                end

                fileName = ['model-Gaussian',featReduction,'Features-',...
                    num2str(features)];
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','usedFreqPeaks','trainingData','mdlAcc',...
                    'AIC','BIC');

            case 'SpectralCluster'
                fileName = ['model-SpectralCluster',featReduction,'Features-',...
                    num2str(features)];
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','usedFreqPeaks','trainingData','mdlAcc');
```

```
        end
    else
        % Train undamaged and damaged models
        if MLmodelType ~= "DL" && MLmodelType ~= "ShallowNN"
            % This case for all conventional ML models

            % Split up data to keep some for testing
            data.Class = knownClass;
            pt = cvpartition(data.Class,"Holdout",0.15);
            dataTrain = data(training(pt),:);
            dataTest = data(test(pt),:);

            % Construct partition to have the highest validation (leaveout)
            cvpt = cvpartition(dataTrain.Class,"leaveout");
            opts = struct("CVPartition",cvpt,"Verbose",1);
        end

        switch MLmodelType
            case 'AUTO'
                % Automatic selection of the best model
                % It takes a very long time to optimise
                mdl = fitcauto(dataTrain,"Class","Learners","auto",...
                    "OptimizeHyperparameters",'auto',...
                    "HyperparameterOptimizationOptions",opts);

                if contains('compact',class(mdl))
                    mdlLoss = 100 * loss(mdl,dataTrain);
                else
                    mdlLoss = 100 * resubLoss(mdl);
                end

                % Info about the model
                [classPred,~] = predict(mdl,dataTest);
                classTrue = dataTest.Class;
                mdlAcc = 100 * ...
                    (sum(nnz(classPred == classTrue))/length(classTrue));

                fileName = ['model-AUTO',featReduction,'Features-',...
                    num2str(features)];
                trainingData = "UD + D";
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','PCA','usedFreqPeaks','trainingData','mdlAcc');

            case 'Tree'
                % Binary tree classification model
                mdl = fitctree(dataTrain,"Class","OptimizeHyperparameters",...
                    "all","HyperparameterOptimizationOptions",opts);

                % Info about the model
                mdlLoss = 100 * resubLoss(mdl);
                [classPred,~] = predict(mdl,dataTest);
                classTrue = dataTest.Class;
                mdlAcc = 100 * ...
                    (sum(nnz(classPred == classTrue))/length(classTrue));

                fileName = ['model-Tree',featReduction,'Features-',...
                    num2str(features)];
                trainingData = "UD + D";
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','PCA','usedFreqPeaks','trainingData','mdlAcc');

            case 'kNN'
                % Nearest neighbors model
                % Categorises points on their distance to points in training data
                mdl = fitcknn(dataTrain,"Class","OptimizeHyperparameters",...
                    "all","HyperparameterOptimizationOptions",opts);

                % Info about the model
```

```
                mdlLoss = 100 * resubLoss(mdl);
                [classPred,~] = predict(mdl,dataTest);
                classTrue = dataTest.Class;
                mdlAcc = 100 * ...
                    (sum(nnz(classPred == classTrue))/length(classTrue));

                fileName = ['model-kNN',featReduction,'Features-',...
                    num2str(features)];
                trainingData = "UD + D";
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','PCA','usedFreqPeaks','trainingData','mdlAcc');

            case 'SVM'
                % Support vector machine model
                % Only usable for 2 distinct class problems
                % Need data to be normalised (mean at 0) so will not be used
                mdl = fitcsvm(dataTrain,"Class","OptimizeHyperparameters",...
                    "all","HyperparameterOptimizationOptions",opts);
                mdl = fitPosterior(mdl);

                % Info about the model
                mdlLoss = 100 * resubLoss(mdl);
                [classPred,~] = predict(mdl,dataTest);
                classTrue = dataTest.Class;
                mdlAcc = 100 * ...
                    (sum(nnz(classPred == classTrue))/length(classTrue));

                fileName = ['model-SVM',featReduction,'Features-',...
                    num2str(features)];
                trainingData = "UD + D";
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','PCA','usedFreqPeaks','trainingData','mdlAcc');

            case 'NB'
                % Naive Bayes model
                % Assumes predictors are independent of each other
                mdl = fitcnb(dataTrain,"Class","OptimizeHyperparameters",...
                    "all","HyperparameterOptimizationOptions",opts);

                % Info about the model
                mdlLoss = 100 * resubLoss(mdl);
                [classPred,~] = predict(mdl,dataTest);
                classTrue = dataTest.Class;
                mdlAcc = 100 * ...
                    (sum(nnz(classPred == classTrue))/length(classTrue));

                fileName = ['model-NB',featReduction,'Features-',...
                    num2str(features)];
                trainingData = "UD + D";
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','PCA','usedFreqPeaks','trainingData','mdlAcc');

            case 'DA'
                % Discriminant analysis classification model
                % Assumes different classes generate data based on Gaussian
distribution
                mdl = fitcdiscr(dataTrain,"Class","OptimizeHyperparameters",...
                    "all","HyperparameterOptimizationOptions",opts);

                % Info about the model
                mdlLoss = 100 * resubLoss(mdl);
                [classPred,~] = predict(mdl,dataTest);
                classTrue = dataTest.Class;
                mdlAcc = 100 * ...
                    (sum(nnz(classPred == classTrue))/length(classTrue));

                fileName = ['model-DA',featReduction,'Features-',...
                    num2str(features)];
```

```
                trainingData = "UD + D";
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','PCA','usedFreqPeaks','trainingData','mdlAcc');

            case 'NN'
                % This is only in MATLAB 2021 so implementing now
%                 % Classification neural network
%
%                 % Construct and train the model
%                 mdl = fitcnet(dataTrain,"Class","Leaveout",'on',...
%                     "Standardize",true);
%
%                 mdlLoss = 100 * resubLoss(mdl);
%                 [classPred,conf] = predict(mdl,dataTest);
%                 classTrue = dataTest.Class;
%                 mdlAcc = 100 * ...
%                    (sum(nnz(classPred == classTrue))/length(classTrue));
%
%                 % Info about the model
%                 fileName = ['model-NN',featReduction,'Features-',...
%                    num2str(features)];
%                 trainingData = "UD + D";
%                 save(fullfile(applicationPath,fileName),'MLmodelType',...
%                     'mdl','PCA','usedFreqPeaks','trainingData','mdlAcc');

            case 'ShallowNN'
                % The neural network case
                net = patternnet(15); % Number of neurons per layer

                % Split up features into separate DS
                net.divideParam.trainRatio = 70/100;
                net.divideParam.valRatio = 15/100;
                net.divideParam.testRatio = 15/100;

                % Format data and true class
                data = table2array(data)';
                classes = dummyvar(knownClass)';

                % Format used to analyse which class is actually is
                [~,classIdx] = max(classes(:,1));
                format = struct(string(knownClass(1)),classIdx);

                % Train the network
                [net,tr] = train(net,data,classes);

                % Info about the network
                mdlLoss = 100 * tr.best_tperf;
                scoreTest = net(data(:,tr.testInd));
                [~,classPred] = max(scoreTest);
                classTrue = double(knownClass(tr.testInd));
                mdlAcc = 100 * ...
                    sum(nnz(classPred' == classTrue))/length(classTrue);

                mdl = net;
                fileName = ['model-ShallowNN',featReduction,'Features-',...
                    num2str(features)];
                trainingData = "UD + D";
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','PCA','format','usedFreqPeaks','trainingData',...
                    'mdlAcc');

            case 'DL'
                % The deep learning case
                % It takes a very long time to train

                [trainImgs,valImgs,testImgs] = splitEachLabel(im_ds,...
                    0.7,0.15,0.15, "randomized");
```

```
                % Split up images into separate DS
                train_ds = augmentedImageDatastore([224 224],trainImgs);
                val_ds = augmentedImageDatastore([224 224],valImgs);
                test_ds = augmentedImageDatastore([224 224],testImgs);

                % https://www.mathworks.com/help/deeplearning/ug/
                % pretrained-convolutional-neural-networks.html
                % Can change the pretrained net that is used

                net = resnet50; % Must have input 224x224

                % Transfer learning
                if isa(net,'SeriesNetwork')
                    lgraph = layerGraph(net.Layers);
                else
                    lgraph = layerGraph(net);
                end

                % Layers to replace
                [learnableLayer,classLayer] = findLayersToReplace(lgraph);

                numClasses = numel(categories(trainImgs.Labels));
                if isa(learnableLayer,'nnet.cnn.layer.FullyConnectedLayer')
                    newLearnableLayer = fullyConnectedLayer(numClasses,...
                        'Name','new_fc');

                elseif isa(learnableLayer,'nnet.cnn.layer.Convolution2DLayer')
                    newLearnableLayer = convolution2dLayer(1,numClasses,...
                        'Name','new_conv');
                end
                lgraph =
replaceLayer(lgraph,learnableLayer.Name,newLearnableLayer);

                newClassLayer = classificationLayer('Name','new_classoutput');
                lgraph = replaceLayer(lgraph,classLayer.Name,newClassLayer);

                % Training options (these have not be optimised and are not
                % optimised automatically)
                options = trainingOptions("adam", ...
                    "Plots","training-progress", ...
                    "ValidationData",val_ds,...
                    "ValidationFrequency",5,...
                    "ValidationPatience",5,...
                    "LearnRateSchedule","piecewise",...
                    "LearnRateDropPeriod",2,...
                    "MaxEpochs",10);

                % Train the network
                [net,info] = trainNetwork(train_ds,lgraph,options);

                % Info about the network
                mdlLoss = 100 * info.TrainingLoss(end);
                mdlAcc =  info.TrainingAccuracy(end);
                [classPred,~] = classify(net,test_ds);
                classTrue = testImgs.Labels;

                fileName = ['model-DL-','ResNet50'];
                mdl = net;
                usedFreqPeaks = [];
                trainingData = "UD + D";
                save(fullfile(applicationPath,fileName),'MLmodelType',...
                    'mdl','usedFreqPeaks','trainingData','mdlAcc');

            otherwise
                status.error = 1;
                status.desc = 'Non valid ML model has been requested';
        end
    end
```

```
catch ME
    status.error = 1;
    status.desc = append('Error in CreateTrainMLAlgorithm.m: ',ME.message);
end

end
```

Appendix I – Classify Function Description and Code

Classify.m was created to use the trained ML model with new test data to identify if the unknown blade was undamaged or damaged. The input was the ML model and the feature extracted data from the blade. The output was the damage condition of the blade with the probability of it being undamaged, if a supervised ML model was used.

```
function [classPred,probability] = Classify(MLmodel,data)
% This function classifies the unknown blade under test

% Function inputs:
% MLmodel - Structure containing the machine learning model to be used
% data - The test data to be compared to the machine learning model

% Function outputs:
% classPred - The predicted classification ('UD' or 'D')
% probability - The probability the blade is 'UD'

% ========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% ------------------------------------------------------------------------

% Classify the blade depending on training dat used and model type
% This is where new models can be added in the switch case structure
if MLmodel.trainingData == "UD only"
    % Classify undamaged only models
    testData = table2array(data);
    trainData = table2array(MLmodel.mdl);
    inputData = trainData;
    inputData(end+1,:) = testData;

    switch MLmodel.MLmodelType
        case "Basic"
            mu = mean(trainData,1);
            stdev = std(trainData);

            for i = 1:length(mu)
                diff(i) = abs(mu(i) - testData(i));
                numStds(i) = diff(i) / stdev(i);
                if numStds(i) > 6
                    % Prevent very large numbers if std is very small
                    numStds(i) = 6;
                end
            end

            meanNumStds = mean(numStds);

            if meanNumStds > 3
                % 3 standard deviations are considered to be an outlier
                classPred = "D";
            else
                classPred = "UD";
            end
            probability = "NA";

        case "Gaussian"
            %https://www.mathworks.com/help/stats/
            %clustering-using-gaussian-mixture-models.html
            GMModels = fitgmdist(inputData,1,'RegularizationValue',0.01);
            AIC = GMModels.AIC;

            % If new file increases AIC by 10% then outlier
            if MLmodel.AIC(1) > 0
```

```
                threshold = 1.1 * MLmodel.AIC(1);
            else
                threshold = 0.9 * MLmodel.AIC(1);
            end

            if AIC > threshold
                classPred = "D";
            else
                classPred = "UD";
            end
            probability = "NA";

        case "DBSCAN"
            %https://www.mathworks.com/help/stats/dbscan.html
            % Give a new file 10% margin on becoming an outlier
            epsilon = round(1.1 * MLmodel.epsilon);
            minpts = round(1.1 * MLmodel.minpts);

            classes = dbscan(inputData,epsilon,minpts);

            % If there is an outlier then there will be a -1 in classes
            if ismember(-1,classes)
                classPred = "D";
            else
                classPred = "UD";
            end
            probability = "NA";

            case "SpectralCluster"
            %https://www.mathworks.com/help/stats/
            %partition-data-using-spectral-clustering.html

            pause(1); % Need to pause or it crashes sometimes
            [~,~,eigenValues] = spectralcluster(inputData,7,...
                'ClusterMethod','kmedoids','NumNeighbors',10);

            % Eigenvalues close to zero represent a cluster
            classes = sum(eigenValues < 1e-5);

            if classes > 1
                classPred = "D";
            else
                classPred = "UD";
            end
            probability = "NA";
    end

else
    % Classify undamaged and damaged models
    switch MLmodel.MLmodelType
        case "DL"
            % Classify using the deep learning case
            [classPred,confidence] = classify(MLmodel.mdl,data);
            if classPred == categorical("UD")
                probability = 100 * max(confidence);
            else
                probability = 100 * min(confidence);
            end

        case "ShallowNN"
            % Classify using the shallow neural network

            if isfield(MLmodel.PCA,'coeff')
                % Transform data if a PCA was used
                % https://www.mathworks.com/help/stats/pca.html
                data = (table2array(data) - MLmodel.PCA.mu) * ...
                    MLmodel.PCA.coeff(:,1:MLmodel.PCA.index);
            else
```

```
                data = table2array(data);
            end

            % Logic below is required because the shallowNN is a special case
            scoreTest = MLmodel.mdl(data');
            [~,classPredScore] = max(scoreTest);

            classType = string(fieldnames(MLmodel.format));
            classIdxConversion = MLmodel.format.(classType);

            if classType == "D"
                classTypeOther = "UD";
            elseif classType == "UD"
                classTypeOther = "D";
            end

            if classPredScore == classIdxConversion
                classPred = categorical(classType);
            else
                classPred = categorical(classTypeOther);
            end

            if (classPred == categorical("UD") && classType == "UD") ||...
                    (classPred == categorical("D") && classType == "UD")
                probability = 100 * scoreTest(classIdxConversion);
            elseif (classPred == categorical("UD") && classType == "D") ||...
                    (classPred == categorical("D") && classType == "D")
                if classIdxConversion == 1
                    probability = 100 * scoreTest(2);
                elseif classIdxConversion == 2
                    probability = 100 * scoreTest(1);
                end
            end

        otherwise
            % Classify using All normal conventional ML methods
            if isfield(MLmodel.PCA,'coeff')
                % Transform data if a PCA was used
                data = (table2array(data) - MLmodel.PCA.mu) * ...
                    MLmodel.PCA.coeff(:,1:MLmodel.PCA.index);
            end

            [classPred,confidence] = predict(MLmodel.mdl,data);
            if classPred == categorical("UD")
                probability = 100 * max(confidence);
            else
                probability = 100 * min(confidence);
            end
    end
end
```

Appendix J – CheckHistoricalBladeTests Function Description and Code

CheckHistoricalBladeTests.m was created to inspect the folder of the training data to identify if a blade had already been tested, and if there were some CSV files present. It was essential to make sure no CSV files were overwritten, and therefore the function contained the logic to prevent this from occurring and assigned the correct index to newly saved files.

```
function [highestIdx,bladeFolderPath] = ...
    CheckHistoricalBladeTests(trainingDataFolderPath,testFileName)
% This function identifies the current index value of saved files

% Function inputs:
% trainingDataFolderPath - The path to where the training data is kept
% testFileName - The file name of the test blade

% Function outputs:
% highestIdx - Highest index of previous saved tests
% bladeFolderPath - The folder path of the test blade being saved
% =========================================================================
% Written by Nicholas Thomas
% 25-04-2021
% -------------------------------------------------------------------------

% Check to see if there is a folder with the blade name
bladeCondition = string(extractAfter(testFileName,'_'));
if bladeCondition == "UD"
    bladeFolder = "Undamaged";
    folders = dir(fullfile(trainingDataFolderPath,bladeFolder));
else
    bladeFolder = "Damaged";
    folders = dir(fullfile(trainingDataFolderPath,bladeFolder));
end

bladeFolderPath = fullfile(trainingDataFolderPath,bladeFolder,testFileName);

folderExist = strcmpi({folders.name},{testFileName});
if any(folderExist)
    % The folder already exists
    checkFiles = 1;
else
    % Make the folder and set index to 0
    mkdir(bladeFolderPath);
    checkFiles = 0;
    highestIdx = 0;
end


% Check for previous repeats of blade inside the folder
% Check for previous tests with the blade name
if checkFiles == 1
    files = dir(fullfile(trainingDataFolderPath,bladeFolder,testFileName) +
"/*.csv");
    filesName = {files.name};

    bladeName = string(extractBefore(filesName,'-'));

    if testFileName(end) == '-'
        testFileName = testFileName(1:end-1);
    end

    % Have 1 in the array if there are files already there
    alreadyThere = bladeName == testFileName;
```

```
    if any(alreadyThere)
        indexes = str2double(extractBetween(filesName(alreadyThere),'-','.'));
        highestIdx = max(indexes);
    else
        highestIdx = 0;
    end
end

end
```

Appendix K – findLayersToRepalce Function Description and Code

findLayersToRepalce.m was copied from the MathWorks website to automatically replace the required layers of a DL network.

```
% From MATHWORKS
% findLayersToReplace(lgraph) finds the single classification layer and the
% preceding learnable (fully connected or convolutional) layer of the layer
% graph lgraph.
function [learnableLayer,classLayer] = findLayersToReplace(lgraph)

if ~isa(lgraph,'nnet.cnn.LayerGraph')
    error('Argument must be a LayerGraph object.')
end

% Get source, destination, and layer names.
src = string(lgraph.Connections.Source);
dst = string(lgraph.Connections.Destination);
layerNames = string({lgraph.Layers.Name}');

% Find the classification layer. The layer graph must have a single
% classification layer.
isClassificationLayer = arrayfun(@(l) ...

(isa(l,'nnet.cnn.layer.ClassificationOutputLayer')|isa(l,'nnet.layer.Classification
Layer')), ...
    lgraph.Layers);

if sum(isClassificationLayer) ~= 1
    error('Layer graph must have a single classification layer.')
end
classLayer = lgraph.Layers(isClassificationLayer);


% Traverse the layer graph in reverse starting from the classification
% layer. If the network branches, throw an error.
currentLayerIdx = find(isClassificationLayer);
while true

    if numel(currentLayerIdx) ~= 1
        error('Layer graph must have a single learnable layer preceding the
classification layer.')
    end

    currentLayerType = class(lgraph.Layers(currentLayerIdx));
    isLearnableLayer = ismember(currentLayerType, ...

['nnet.cnn.layer.FullyConnectedLayer','nnet.cnn.layer.Convolution2DLayer']);

    if isLearnableLayer
        learnableLayer =  lgraph.Layers(currentLayerIdx);
        return
    end

    currentDstIdx = find(layerNames(currentLayerIdx) == dst);
    currentLayerIdx = find(src(currentDstIdx) == layerNames);

end

end
```

Appendix L – MATLAB GUI Application Code

```
classdef Pass_Fail_App_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        TabGroup                        matlab.ui.container.TabGroup
        MainPassFailApplicationTab      matlab.ui.container.Tab
        MachineLearningSetupPanel       matlab.ui.container.Panel
        ModelDropDownLabel              matlab.ui.control.Label
        MLModelDropDown                 matlab.ui.control.DropDown
        TrainedLabel                    matlab.ui.control.Label
        TypeLabel                       matlab.ui.control.Label
        AccuracyLabel                   matlab.ui.control.Label
        TypeLabel_2                     matlab.ui.control.Label
        AccuracyLabel_2                 matlab.ui.control.Label
        TrainedLabel_2                  matlab.ui.control.Label
        ResultsPanel                    matlab.ui.container.Panel
        PredictedBladeCondition         matlab.ui.control.Lamp
        BladeConditionLabel             matlab.ui.control.Label
        PredictedClass                  matlab.ui.control.Label
        ProbabilityBladeCondition       matlab.ui.control.Label
        TestSetupPanel                  matlab.ui.container.Panel
        TestTypeButtonGroup             matlab.ui.container.ButtonGroup
        LiveButton                      matlab.ui.control.RadioButton
        SimulatedButton                 matlab.ui.control.RadioButton
        BrowseButtonFile                matlab.ui.control.Button
        TheSimulatedTestFileLabel       matlab.ui.control.Label
        SelectedTestFileLabel           matlab.ui.control.Label
        StartButtonMLTest               matlab.ui.control.Button
        PlotOptionsPanel                matlab.ui.container.Panel
        HoldOnPlotsSwitchLabel          matlab.ui.control.Label
        HoldOnPlotsSwitch               matlab.ui.control.Switch
        FrequencyMaxkHzSpinnerLabel     matlab.ui.control.Label
        FrequencyMaxkHzSpinner          matlab.ui.control.Spinner
        StopButtonMLTest                matlab.ui.control.Button
        ProgressMLTest                  matlab.ui.control.Button
        UIAxesRaw_Main                  matlab.ui.control.UIAxes
        UIAxesFFTAmp                    matlab.ui.control.UIAxes
        CollectMoreTrainingDataTab      matlab.ui.container.Tab
        TestSetupPanel_2                matlab.ui.container.Panel
        BrowseButtonFolderTestData      matlab.ui.control.Button
        TheBladesTopLevelFolderPathLabel_3  matlab.ui.control.Label
        SelectedFolderPathLabel         matlab.ui.control.Label
        FileNameEditFieldLabel          matlab.ui.control.Label
        FileNameEditField               matlab.ui.control.EditField
        NumberofTestsSpinnerLabel       matlab.ui.control.Label
        NumberofTestsSpinner            matlab.ui.control.Spinner
        StartButtonTestData             matlab.ui.control.Button
        StopButtonTestData              matlab.ui.control.Button
        ProgressTestData                matlab.ui.control.Button
        UIAxesRaw_Col                   matlab.ui.control.UIAxes
        ManualFeatureExtractionTab      matlab.ui.container.Tab
        StartButtonFeatExtract          matlab.ui.control.Button
        StopButtonFeatExtract           matlab.ui.control.Button
        NextFileButtonFeatExtract       matlab.ui.control.Button
        RandomFileLabelFeatExtract      matlab.ui.control.Label
        SelectedFeatExtract             matlab.ui.control.Label
        DeleteRowButtonFeatExtract      matlab.ui.control.Button
        ManualFeatureExtractionSetupPanel  matlab.ui.container.Panel
        BrowseButtonFolderReML          matlab.ui.control.Button
        TheBladesTopLevelFolderPathLabel  matlab.ui.control.Label
        SelectedFolderPathLabelFeatExtract  matlab.ui.control.Label
        ExtractedfeaturesmatchthemachinelearningalgorithmLabel
matlab.ui.control.Label
        FrequencyMaxkHzSpinner_2Label   matlab.ui.control.Label
```

```
        FrequencyMaxkHzSpinnerFeatExtract  matlab.ui.control.Spinner
        FrequencyMinkHzLabel               matlab.ui.control.Label
        FrequencyMinkHzSpinnerFeatExtract  matlab.ui.control.Spinner
        UITablesFrequencyModes             matlab.ui.control.Table
        RownumberEditFieldLabel            matlab.ui.control.Label
        RowNumberEditFieldFeatExtract      matlab.ui.control.NumericEditField
        ConditionLabelFeatExtract          matlab.ui.control.Label
        BladeConditionSwitchFeatExtract    matlab.ui.control.Switch
        UIAxesFeatExtract                  matlab.ui.control.UIAxes
        RetrainMachineLearningModelTab     matlab.ui.container.Tab
        RetrainMachineLearningModelSetupPanel  matlab.ui.container.Panel
        BrowseButtonFolderReTrainML        matlab.ui.control.Button
        TheBladesTopLevelFolderPathLabel_2  matlab.ui.control.Label
        SelectedFolderPathLabelReTrainML   matlab.ui.control.Label
        TrainingDataAvailableButtonGroup   matlab.ui.container.ButtonGroup
        UndamagedandDamagedButton          matlab.ui.control.RadioButton
        UndamagedButton                    matlab.ui.control.RadioButton
        NumberOfFeaturesText               matlab.ui.control.Label
        NumberOfFeatures                   matlab.ui.control.Label
        MachineLearningMethodtoTrainListBoxLabel  matlab.ui.control.Label
        MachineLearningMethodtoTrainListBox  matlab.ui.control.ListBox
        MLmodelTable                       matlab.ui.control.Table
        ConfusionChartPanel                matlab.ui.container.Panel
        StartButtonReTrainML               matlab.ui.control.Button
        CurrentTrainedModelsLabel_2        matlab.ui.control.Label
        AccuracyofthemodelLabel            matlab.ui.control.Label
        AccuracyOfTheModel                 matlab.ui.control.Label
        LossofthemodelLabel                matlab.ui.control.Label
        LossOfTheModel                     matlab.ui.control.Label
        CurrentTrainedModelsLabel_3        matlab.ui.control.Label
        CurrentTrainedModelsLabel_4        matlab.ui.control.Label
        AutoReduceFeaturesButtonGroup      matlab.ui.container.ButtonGroup
        YesButton                          matlab.ui.control.RadioButton
        NoButton                           matlab.ui.control.RadioButton
        Descp                              matlab.ui.control.Label
        CurrentTrainedModelsLabel_5        matlab.ui.control.Label
        DeleteRowButtonReTrainML           matlab.ui.control.Button
        ModeltoDeleteDropDownLabel         matlab.ui.control.Label
        ModeltoDeleteDropDown              matlab.ui.control.DropDown
        StatusLabel                        matlab.ui.control.Label
        ErrorLamp                          matlab.ui.control.Lamp
    end


    properties (Access = private)
        applicationPath % The path the application is in
        MLmodel % The ML mdl and params used to train it and to be used in the test
        featFileCreateDate % Date feature extraction occurred
        MLFileCreateDate % Date ML was trained
        CSVFilePath % The file path and name of a CSV file
        trainingDataFolderPath % The folder path for the training data
        prevFreqPeaks % The saved frequency peaks from the .mat file
        stopPressed % 0 or 1 if a stop button has been pressed
    end

    methods (Access = private)


        function CheckRequiredToolbox(app)
            % This function checks if all the required toolboxes are installed
            requiredToolBoxes = {'Parallel Computing Toolbox',...
                'Signal Processing Toolbox',...
                'Statistics and Machine Learning Toolbox',...
                'Instrument Control Toolbox',...
                'PicoScope Support Toolbox'};
            v_=ver;
            [installedToolboxes{1:length(v_)}] = deal(v_.Name);
            for i = 1:length(requiredToolBoxes)
```

```
            result = all(ismember(requiredToolBoxes(i),installedToolboxes));
            if result == 0
                f = uifigure;
                msg = ['You have a missing toolbox which must be ',...
                    'installed to run the application. ',...
                    'Missing Toolbox: ',requiredToolBoxes(i)];
                title = 'Missing Toolbox';
                uiconfirm(f,msg,title,...
                   'Options',{'Okay'},...
                   'DefaultOption',1,'CancelOption',1,...
                   'CloseFcn',@(h,e) close(f));
            end
        end
    end

    function PopulateMLDropDownOptions(app)
        % This function updates the ML drop down option box and desc
        try
            models = dir(fullfile(app.applicationPath,'model*.mat'));
            for i = 1:numel(models)
                modelFile = models(i);
                modelFileName = extractBefore(modelFile.name,'.');
                modelName{i} = extractAfter(modelFileName,'-');
            end
            app.MLModelDropDown.Items = modelName;
            app.ModeltoDeleteDropDown.Items = modelName;

            MLModelDropDownValueChanged(app,[]);
        catch
            UpdateStatus(app,0,"No trained models")
        end
    end

    function UpdateMLTestPlots(app,dataOut,holdOn)
        % Function updates the plots with the new output data
        if holdOn == "Yes"
            hold(app.UIAxesRaw_Main,"on")
            hold(app.UIAxesFFTAmp,"on")
        end

        plot(app.UIAxesRaw_Main,dataOut.rawSigTime,dataOut.rawSig)
        xlim(app.UIAxesRaw_Main,[0 inf])

        plot(app.UIAxesFFTAmp,dataOut.f/1000,dataOut.amp)
        hold(app.UIAxesFFTAmp,"on")
        if app.MLmodel.MLmodelType ~= "DL"
        plot(app.UIAxesFFTAmp,dataOut.extractedFreqs/1000,...
            dataOut.ampAtExtractedFreqs,'r^','markerfacecolor',[1 0 0])
        end
        hold(app.UIAxesFFTAmp,"off")
        xlim(app.UIAxesFFTAmp,[0 app.FrequencyMaxkHzSpinner.Value])
    end

    function ResetMLTestPlots(app)
        % Function clears the plots
        hold(app.UIAxesRaw_Main,"off")
        hold(app.UIAxesFFTAmp,"off")
        xlim(app.UIAxesRaw_Main,[0 inf])
        xlim(app.UIAxesFFTAmp,[0 inf])
        plot(app.UIAxesRaw_Main,0,0);
        plot(app.UIAxesFFTAmp,0,0);
    end

    function PopulateFrequencyModesTable(app)
        % Function populates the table of mode frequencies
        numNodes = length(app.prevFreqPeaks.freqPeaks);
        if numNodes ~= 0
            app.UITablesFrequencyModes.Data = [string(1:numNodes)',...
```

```
                    string(app.prevFreqPeaks.freqPeaks)'];
            else
                app.UITablesFrequencyModes.Data = [-1,-1];
            end
        end

        function UpdateFeatExtractPlots(app)
            % Function plots the file and peaks from .mat file
            data = readtable(app.CSVFilePath);
            dataOut = PreprocessData(data);
            plot(app.UIAxesFeatExtract,dataOut.f/1000,dataOut.amp)
            xlim(app.UIAxesFeatExtract,...
                [app.FrequencyMinkHzSpinnerFeatExtract.Value
app.FrequencyMaxkHzSpinnerFeatExtract.Value])

            freqPeaks = app.prevFreqPeaks.freqPeaks;
            for i = 1:length(freqPeaks)
                if ~isnan(freqPeaks(i))
                    xline(app.UIAxesFeatExtract,freqPeaks(i),'--r');
                end
            end
        end

        function PopulateReTrainMLTable(app)
            % This function populates the table of previous ML models
            try
                models = dir(fullfile(app.applicationPath,'model*.mat'));
                for i = 1:numel(models)
                    modelFile = models(i);
                    modelFileName = extractBefore(modelFile.name,'.');
                    modelName = extractAfter(modelFileName,'-');
                    modelFileTrained = extractBefore(modelFile.date,' ');
                    modelInfo = load(modelFileName);

                    modelType = modelInfo.trainingData;
                    modelAcc = modelInfo.mdlAcc;

                    data(i,:) =
[modelName,modelFileTrained,modelType,string(modelAcc)];
                end
                set(app.MLmodelTable,'Data',data,...
                    'ColumnFormat',{'char','char','char','char'})
            catch
                UpdateStatus(app,0,"No trained models")
            end
        end

        function UpdateStatus(app,error,message)
            % This function updates the status bar for the progress of the
            % test and of any errors
            app.StatusLabel.Text = message;
            if error == 1
                app.ErrorLamp.Color = 'red';
            else
                app.ErrorLamp.Color = 'green';
            end
        end
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            % Functions to run on start-up of the application
            set(app.StopButtonMLTest,'Enable','off');
            set(app.StartButtonTestData,'Enable','off');
            set(app.StopButtonTestData,'Enable','off');
```

```
            set(app.StartButtonFeatExtract,'Enable','off');
            set(app.StopButtonFeatExtract,'Enable','off');
            set(app.NextFileButtonFeatExtract,'Enable','off');
            set(app.BladeConditionSwitchFeatExtract,'Enable','off');
            set(app.DeleteRowButtonFeatExtract,'Enable','off');
            set(app.StartButtonReTrainML,'Enable','off');

            app.stopPressed = 0;
            app.applicationPath = fileparts(mfilename('fullpath'));

            try
                app.prevFreqPeaks = load(fullfile(app.applicationPath,...
                    'freqPeaksLoc'));
                file = dir(fullfile(app.applicationPath,'freqPeaksLoc.mat'));
                app.featFileCreateDate = extractBefore(file.date,' ');
            catch
                % No .mat file has been created
                errordlg("No features have been identifed and selected yet",...
                    "No .mat file")
                app.prevFreqPeaks.freqPeaks = [];
                app.UITablesFrequencyModes.Data = [-1,-1];
            end

            TestTypeButtonGroupSelectionChanged(app,[]);
            PopulateMLDropDownOptions(app);
            ResetMLTestPlots(app);
            PopulateFrequencyModesTable(app);
            PopulateReTrainMLTable(app);
            TrainingDataAvailableButtonGroupSelectionChanged(app,[]);
            app.NumberOfFeatures.Text =
string(length(app.prevFreqPeaks.freqPeaks));
            CheckRequiredToolbox(app);
            UpdateStatus(app,0,append('Application Folder: ',app.applicationPath));
        end

        % Button pushed function: StopButtonFeatExtract,
        % StopButtonMLTest, StopButtonTestData
        function StopButtonPushed(app, event)
            % This function breaks the code out of any loops and stops the
            % test
            app.stopPressed = 1;
        end

        % Button pushed function: BrowseButtonFile
        function BrowseButtonFileMLPushed(app, event)
            % This function browses the PC to find a simulated signal
            % file to use
            try
                [file,path] = uigetfile('*.csv');
                app.SelectedTestFileLabel.Text = file;
                app.CSVFilePath = fullfile(path,file);
                set(app.StartButtonMLTest,'Enable','on')
            catch
                errordlg("A sample signal .csv file must be selected",...
                    "Invalid selection")
            end
        end

        % Selection changed function: TestTypeButtonGroup
        function TestTypeButtonGroupSelectionChanged(app, event)
            % The function controls if the user can select a simulated
            % data file
            selectedButton = app.TestTypeButtonGroup.SelectedObject.Text;

            if selectedButton == "Live"
                set(app.BrowseButtonFile,'Enable','off')
                set(app.SelectedTestFileLabel,'Enable','off')
                set(app.StartButtonMLTest,'Enable','on')
```

```
        elseif selectedButton == "Simulated"
            set(app.BrowseButtonFile,'Enable','on')
            set(app.SelectedTestFileLabel,'Enable','on')

            if contains(app.SelectedTestFileLabel.Text,".csv")
                set(app.StartButtonMLTest,'Enable','on')
            else
                set(app.StartButtonMLTest,'Enable','off')
            end
        end
    end

    % Value changed function: MLModelDropDown
    function MLModelDropDownValueChanged(app, event)
        % This function updates the information for the selected model
        selectedModel = app.MLModelDropDown.Value;
        selectedModelFileName = append('model-',selectedModel);
        modelFileInfo = dir(fullfile(app.applicationPath,...
            append(selectedModelFileName,'.mat')));

        app.MLmodel = load(selectedModelFileName);
        app.TypeLabel_2.Text = app.MLmodel.trainingData;
        app.AccuracyLabel_2.Text = string(app.MLmodel.mdlAcc);
        app.TrainedLabel_2.Text = extractBefore(modelFileInfo.date,' ');
    end

    % Button pushed function: StartButtonMLTest
    function StartButtonMLTestPushed(app, event)
        % Begin the pass fail test
        set(app.StartButtonMLTest,'Enable','off')
        set(app.BrowseButtonFile,'Enable','off')
        set(app.MLModelDropDown,'Enable','off')
        set(app.TestTypeButtonGroup,'Enable','off')
        set(app.StopButtonMLTest,'Enable','on');

        testType = app.TestTypeButtonGroup.SelectedObject.Text;
        holdOn = app.HoldOnPlotsSwitch.Value;
        ResetMLTestPlots(app);
        app.PredictedClass.Text = "UD/D";
        app.ProbabilityBladeCondition.Text = ...
            ['Probability it is undamaged'];
        app.PredictedBladeCondition.Color = 'white';

        UpdateStatus(app,0,"Testing the Blade");

        % This section creates the loading bar
        originalButtonText = app.ProgressMLTest.Text;
        app.ProgressMLTest.Text = "Collecting...";
        app.ProgressMLTest.IconAlignment = 'bottom';
        wbar = permute(repmat(app.ProgressMLTest.BackgroundColor,...
            30,1,350),[1,3,2]);
        wbar([1,end],:,:) = 0;
        wbar(:,[1,end],:) = 0;
        app.ProgressMLTest.Icon = wbar;
        % Loading bar created

        n = 5;
        for i = 1:n
            % Check for stop button press
            if app.stopPressed == 1
                break
            end

            [dataOut(i),classPred(i),probability(i),status] = ...
                RunSingleTest(app.MLmodel,testType,app.CSVFilePath);

            % Check for errors in the test
```

```
                if status.error == 1
                    UpdateStatus(app,1,status.desc);
                    break
                end

                UpdateMLTestPlots(app,dataOut(i),holdOn);
                app.PredictedClass.Text = classPred(i);

                if app.MLmodel.trainingData == "UD + D"
                    desc = ['The probability that the blade is undamaged: ',...
                        append(string(round(probability(i))),' %')];
                else
                    desc = ['No probability is obtained when using an UD
model.',...
                        ' Check the prediction with the other UD models'];
                end
                app.ProbabilityBladeCondition.Text = desc;

                if string(classPred(i)) == "UD"
                    app.PredictedBladeCondition.Color = 'green';
                else
                    app.PredictedBladeCondition.Color = 'red';
                end

                % This section updates the loading bar
                currentProg = min(round((size(wbar,2)-2)*(i/n)),size(wbar,2)-2);
                app.ProgressMLTest.Text = "Impact " + num2str(i) +...
                    " of " + num2str(n);
                app.ProgressMLTest.Icon(2:end-1, 2:currentProg+1, 1) = 0.25391;
                app.ProgressMLTest.Icon(2:end-1, 2:currentProg+1, 2) = 0.41016;
                app.ProgressMLTest.Icon(2:end-1, 2:currentProg+1, 3) = 0.87891;
                pause(0.25);
                % Loading bar updated
            end

            if app.MLmodel.trainingData == "UD + D"
                if ~(all(classPred == categorical("D")) || all(classPred ==
categorical("UD")))
                    desc = ['Blade classified as both undamaged and damaged',...
                        ' rerun the test'];
                    app.PredictedClass.Text = "Unknown";
                    app.PredictedBladeCondition.Color = 'white';
                else
                    aveProb = mean(probability);
                    desc = ['Average probability that the blade is undamaged: ',...
                        append(string(round(aveProb)),' %')];
                end
                app.ProbabilityBladeCondition.Text = desc;

            end


            hold(app.UIAxesRaw_Main,"off")
            hold(app.UIAxesFFTAmp,"off")

            % Reset the buttons
            app.ProgressMLTest.Icon = '';
            app.ProgressMLTest.Text = originalButtonText;
            set(app.StopButtonMLTest,'Enable','off');
            set(app.StartButtonMLTest,'Enable','on');
            set(app.BrowseButtonFile,'Enable','on')
            set(app.TestTypeButtonGroup,'Enable','on')
            set(app.MLModelDropDown,'Enable','on')

            if status.error == 0
                if app.stopPressed == 0
                    UpdateStatus(app,0,"Test Complete");
                else
```

```
                UpdateStatus(app,0,"Test Stopped");
                app.stopPressed = 0;
            end
        end
    end

    % Button pushed function: BrowseButtonFolderTestData
    function BrowseButtonFolderTestDataPushed(app, event)
        % This function browses the PC to select a save folder path
        try
            app.trainingDataFolderPath = uigetdir();
            app.SelectedFolderPathLabel.Text = app.trainingDataFolderPath;
        catch
            errordlg("A folder location must be selected",...
                "Invalid selection")
        end

        if contains(app.SelectedFolderPathLabel.Text,"Selected Folder Path")
            set(app.StartButtonTestData,'Enable','off')
        else
            set(app.StartButtonTestData,'Enable','on')
        end
    end

    % Button pushed function: StartButtonTestData
    function StartButtonTestDataPushed(app, event)
        % This function begins the collect training data method
        set(app.StartButtonTestData,'Enable','off')
        set(app.StopButtonTestData,'Enable','on');
        set(app.BrowseButtonFolderTestData,'Enable','off');
        set(app.FileNameEditField,'Enable','off');
        set(app.NumberofTestsSpinner,'Enable','off');

        UpdateStatus(app,0,"Collecting Impact Test Data");

        % This section creates the loading bar
        originalButtonText = app.ProgressTestData.Text;
        app.ProgressTestData.Text = "Collecting...";
        app.ProgressTestData.IconAlignment = 'bottom';
        wbar = permute(repmat(app.ProgressTestData.BackgroundColor,...
            30,1,350),[1,3,2]);
        wbar([1,end],:,:) = 0;
        wbar(:,[1,end],:) = 0;
        app.ProgressTestData.Icon = wbar;
        % Loading bar created

        [highestIdx,bladeFolderPath] = CheckHistoricalBladeTests(...
            app.trainingDataFolderPath,app.FileNameEditField.Value);

        n = app.NumberofTestsSpinner.Value;
        for i = 1:n
            if app.stopPressed == 1
                break
            end

            % Do the impact event once
            idx = highestIdx + i;
            [dataOut,status] = ImpactTest("Yes",idx,bladeFolderPath,...
                app.FileNameEditField.Value);

            % Check for errors in the test
            if status.error == 1
                UpdateStatus(app,1,status.desc);
                break
            end

            plot(app.UIAxesFeatExtract,dataOut.Time,dataOut.ChannelA)
            xlim(app.UIAxesFeatExtract,[0 inf])
```

```
                % This section updates the loading bar
                currentProg = min(round((size(wbar,2)-2)*(i/n)),size(wbar,2)-2);
                app.ProgressTestData.Text = "Test " + num2str(i) +...
                    " of " + num2str(n);
                app.ProgressTestData.Icon(2:end-1, 2:currentProg+1, 1) = 0.25391;
                app.ProgressTestData.Icon(2:end-1, 2:currentProg+1, 2) = 0.41016;
                app.ProgressTestData.Icon(2:end-1, 2:currentProg+1, 3) = 0.87891;
                pause(0.25);
                % Loading bar updated
            end

            % Reset the buttons
            app.ProgressTestData.Icon = '';
            app.ProgressTestData.Text = originalButtonText;
            set(app.StopButtonTestData,'Enable','off');
            set(app.StartButtonTestData,'Enable','on');
            set(app.BrowseButtonFolderTestData,'Enable','on');
            set(app.FileNameEditField,'Enable','on');
            set(app.NumberofTestsSpinner,'Enable','on');

            if app.stopPressed == 0
                UpdateStatus(app,0,"Test Data Collected");
            else
                UpdateStatus(app,0,"Test Stopped");
                app.stopPressed = 0;
            end
        end

        % Button pushed function: BrowseButtonFolderReML
        function BrowseButtonFolderFeatExtractPushed(app, event)
            % This function browses the PC to select a save folder path
            try
                app.trainingDataFolderPath = uigetdir();
                app.SelectedFolderPathLabelFeatExtract.Text =
app.trainingDataFolderPath;
            catch
                errordlg("A folder location must be selected",...
                    "Invalid selection")
            end

            if ~contains(app.SelectedFolderPathLabelFeatExtract.Text,...
                    "Selected Folder Path")
                set(app.StartButtonFeatExtract,'Enable','on')
                set(app.BladeConditionSwitchFeatExtract,'Enable','on');
                set(app.NextFileButtonFeatExtract,'Enable','on')
            end
        end

        % Button pushed function: StartButtonFeatExtract
        function StartButtonReTrainPeaksPushed(app, event)
            % This function begins the feature selection process
            set(app.StartButtonFeatExtract,'Enable','off');
            set(app.NextFileButtonFeatExtract,'Enable','off');
            set(app.BladeConditionSwitchFeatExtract,'Enable','off');
            set(app.StopButtonFeatExtract,'Enable','on');
            set(app.DeleteRowButtonFeatExtract,'Enable','on');
            UpdateStatus(app,0,"Collecting features");

            try
                app.prevFreqPeaks = load(fullfile(app.applicationPath,...
                    'freqPeaksLoc'));
            catch
                % No .mat file has been created
                app.prevFreqPeaks.freqPeaks = [];
            end

            if contains(app.SelectedFeatExtract.Text,".csv")
```

```
                % Select current file and plot it with .mat file peaks
                UpdateFeatExtractPlots(app);
            else
                % Select new file and plot it with .mat file peaks
                NextFileButtonPushed(app);
            end

            changes = 0;
            while app.stopPressed == 0
                try
                dataTip = findall(app.UIAxesFeatExtract,'Type','hggroup');
                pause(0.2)
                if isempty(dataTip)
                    % No data tip has been selected
                else
                    try
                        pause(0.2) % Important, without it Matlab crashes
                        coords = dataTip.Position;
                    catch
                        pause(1) % Important, without it Matlab crashes
                        coords = dataTip.Position;
                    end
                    newFreq = coords(1);
                    app.prevFreqPeaks.freqPeaks = sort(cat(2,newFreq,...
                        app.prevFreqPeaks.freqPeaks),"ascend");

                    % Update the table with the new frequency
                    PopulateFrequencyModesTable(app);

                    % Update the graph with the new frequency
                    delete(dataTip);
                    xline(app.UIAxesFeatExtract,newFreq,'--r');
                    changes = changes + 1;
                end
                catch
                    % error on findall function
                end
            end

            if changes > 0
                f = uifigure;
                msg = ['There are new values, do you want to save the changes?
',...
                    'To use the new features selected, a new machine ',...
                    'learning model will need to be trained'];
                title = 'Confirm Save';
                selection = uiconfirm(f,msg,title,...
                           'Options',{'Save','Cancel'},...
                           'DefaultOption',2,'CancelOption',2,...
                           'CloseFcn',@(h,e) close(f));

                if selection == "Save"
                    freqPeaks = sort(app.prevFreqPeaks.freqPeaks,"ascend");
                    save(fullfile(app.applicationPath,"freqPeaksLoc"),...
                        'freqPeaks')
                    app.NumberOfFeatures.Text = string(length(freqPeaks));
                else
                    % Load previous file and update plot and table
                    try
                        app.prevFreqPeaks = load(fullfile(app.applicationPath,...
                            'freqPeaksLoc'));
                    catch
                        % No .mat file has been created
                        app.prevFreqPeaks.freqPeaks = [];
                    end
                    UpdateFeatExtractPlots(app)
                    PopulateFrequencyModesTable(app);
                end
```

```
            end

            app.stopPressed = 0;

            UpdateStatus(app,0,"Features Collected");
            set(app.StartButtonFeatExtract,'Enable','on');
            set(app.NextFileButtonFeatExtract,'Enable','on');
            set(app.BladeConditionSwitchFeatExtract,'Enable','on');
            set(app.StopButtonFeatExtract,'Enable','off');
        end

        % Callback function: BladeConditionSwitchFeatExtract,
        % NextFileButtonFeatExtract
        function NextFileButtonPushed(app, event)
            % This function choses a random file from the selected folder
            bladeCondition = app.BladeConditionSwitchFeatExtract.Value;
            fileList =
dir(fullfile(app.trainingDataFolderPath,bladeCondition,'**/*.csv'));
            fileIdx = randi(length(fileList));
            fileName = fileList(fileIdx).name;
            filePath = fileList(fileIdx).folder;
            app.SelectedFeatExtract.Text = fileName;
            app.CSVFilePath = fullfile(filePath,fileName);

            UpdateFeatExtractPlots(app);
            set(app.DeleteRowButtonFeatExtract,'Enable','on')
        end

        % Button pushed function: DeleteRowButtonFeatExtract
        function DeleteRowButtonFeatExtractPushed(app, event)
            % This button deletes a row from the frequency peak table and
            % updates the .mat file and re plots the lines
            f = uifigure;
            msg = ['Do you want to delete the selected peak? To use the '...
               'new features selected, a new machine learning model ',...
               'will need to be trained'];
            title = 'Confirm Delete';
            selection = uiconfirm(f,msg,title,...
                       'Options',{'Delete','Cancel'},...
                       'DefaultOption',2,'CancelOption',2,...
                       'CloseFcn',@(h,e) close(f));

            if selection == "Delete"
                rowNum = app.RowNumberEditFieldFeatExtract.Value;
                try
                    app.UITablesFrequencyModes.Data(rowNum,:) = [];
                    app.UITablesFrequencyModes.Data(:,1) = string(1:...
                        length(app.UITablesFrequencyModes.Data(:,1)));
                catch
                    errordlg("Row number is not valid",...
                        "Invalid selection");
                end

                app.prevFreqPeaks.freqPeaks =
str2double(app.UITablesFrequencyModes.Data(:,2))';
                freqPeaks = sort(app.prevFreqPeaks.freqPeaks,"ascend");
                save(fullfile(app.applicationPath,"freqPeaksLoc"),...
                    'freqPeaks')
                app.NumberOfFeatures.Text = string(length(freqPeaks));

                PopulateFrequencyModesTable(app);
                UpdateFeatExtractPlots(app);
            end
        end

        % Value changed function:
        % FrequencyMaxkHzSpinnerFeatExtract,
        % FrequencyMinkHzSpinnerFeatExtract
```

```
        function FrequencyMaxkHzSpinnerFeatExtractValueChanged(app, event)
            % This function changes the x axis of the Feature Extraction
            % plot when the value is changed
            minValue = app.FrequencyMinkHzSpinnerFeatExtract.Value;
            maxValue = app.FrequencyMaxkHzSpinnerFeatExtract.Value;
            xlim(app.UIAxesFeatExtract,[minValue maxValue]);
        end

        % Button pushed function: BrowseButtonFolderReTrainML
        function BrowseButtonFolderReTrainMLPushed(app, event)
            % This function browses the PC to select a save folder path
            try
                app.trainingDataFolderPath = uigetdir();
                app.SelectedFolderPathLabelReTrainML.Text =
app.trainingDataFolderPath;
            catch
                errordlg("A folder location must be selected",...
                    "Invalid selection")
            end

            if ~contains(app.SelectedFolderPathLabelReTrainML.Text,...
                    "Selected Folder Path")
                set(app.StartButtonReTrainML,'Enable','on')
            end
        end

        % Selection changed function:
        % TrainingDataAvailableButtonGroup
        function TrainingDataAvailableButtonGroupSelectionChanged(app, event)
            % This function updates the possible ML models to train
            % depending on the training data
            trainingDataType =
app.TrainingDataAvailableButtonGroup.SelectedObject.Text;

            if trainingDataType == "Undamaged"
                app.MachineLearningMethodtoTrainListBox.Items = ...
                    {'Basic','Gaussian','DBSCAN'};
                set(app.AutoReduceFeaturesButtonGroup,'Enable','off');
            else
                app.MachineLearningMethodtoTrainListBox.Items = ...
                    {'AUTO','Tree','kNN','NB','DA','ShallowNN','DL'};
                set(app.AutoReduceFeaturesButtonGroup,'Enable','on');
            end
        end

        % Button pushed function: StartButtonReTrainML
        function StartButtonReTrainMLPushed(app, event)
            % This function calls the start training functions
            if app.NumberOfFeatures.Text == "0"
                UpdateStatus(app,1,['No peaks have been extracted yet, '...
                    'use the Manual Feature Extraction tab']);
                return
            end

            set(app.StartButtonReTrainML,'Enable','off');
            set(app.BrowseButtonFolderReTrainML,'Enable','off');
            set(app.TrainingDataAvailableButtonGroup,'Enable','off');
            set(app.AutoReduceFeaturesButtonGroup,'Enable','off');

            f = uifigure;
            msg = ['Training of the models cannot be stopped after they',...
                ' have begun. DL and AUTO can take a very long time to train',...
                '. The application will need to be restarted if the training',...
                ' is manually tried to be stopped.',...
                ' The progress pop up takes a number of seconds to load'];
            title = 'Begin Training?';
            selection = uiconfirm(f,msg,title,...
                       'Options',{'Yes','Cancel'},...
```

```
                        'DefaultOption',2,'CancelOption',2,...
                        'CloseFcn',@(h,e) close(f));

            if selection == "Yes"
                UpdateStatus(app,0,"Training begun - monitor progress in pop up");

                [mdlAcc,mdlLoss,classTrue,classPred,~,status] = ...
                    CreateTrainMLAlgorithm(...
                    app.MachineLearningMethodtoTrainListBox.Value,...
                    app.TrainingDataAvailableButtonGroup.SelectedObject.Text,...
                    app.AutoReduceFeaturesButtonGroup.SelectedObject.Text,...
                    app.prevFreqPeaks.freqPeaks,...
                    app.trainingDataFolderPath,...
                    app.applicationPath);

                % Check for errors in the test
                if status.error == 1
                    UpdateStatus(app,1,status.desc);
                else
                    app.AccuracyOfTheModel.Text = append(string(mdlAcc),' %');
                    app.LossOfTheModel.Text = append(string(mdlLoss),' %');
                    if ~isempty(classTrue)

confusionchart(app.ConfusionChartPanel,classTrue,classPred);
                    else
                        delete(app.ConfusionChartPanel.Children)
                    end

                    PopulateReTrainMLTable(app)
                    PopulateMLDropDownOptions(app);
                    TrainingDataAvailableButtonGroupSelectionChanged(app,[]);
                    UpdateStatus(app,0,"Training Completed");
                end
            end

            set(app.StartButtonReTrainML,'Enable','on');
            set(app.BrowseButtonFolderReTrainML,'Enable','on');
            set(app.TrainingDataAvailableButtonGroup,'Enable','on');
            set(app.AutoReduceFeaturesButtonGroup,'Enable','on');
        end

        % Button pushed function: DeleteRowButtonReTrainML
        function DeleteRowButtonReTrainMLPushed(app, event)
            % This function deletes the selected model
            f = uifigure;
            msg = ['Selecting yes will delete the select model.',...
                ' The model will need to be trained again to use it'];
            title = 'Delete Model?';
            selection = uiconfirm(f,msg,title,...
                        'Options',{'Yes','Cancel'},...
                        'DefaultOption',2,'CancelOption',2,...
                        'CloseFcn',@(h,e) close(f));

            if selection == "Yes"
                modelToDel = app.ModeltoDeleteDropDown.Value;
                modelFullFileName = append('model-',modelToDel,'.mat');
                delete(fullfile(app.applicationPath,modelFullFileName));
                PopulateReTrainMLTable(app);
                PopulateMLDropDownOptions(app);
            end

        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
```

```
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 1228 782];
            app.UIFigure.Name = 'MATLAB App';

            % Create TabGroup
            app.TabGroup = uitabgroup(app.UIFigure);
            app.TabGroup.Position = [1 30 1230 753];

            % Create MainPassFailApplicationTab
            app.MainPassFailApplicationTab = uitab(app.TabGroup);
            app.MainPassFailApplicationTab.Title = 'Main Pass/Fail Application';

            % Create MachineLearningSetupPanel
            app.MachineLearningSetupPanel =
uipanel(app.MainPassFailApplicationTab);
            app.MachineLearningSetupPanel.TitlePosition = 'centertop';
            app.MachineLearningSetupPanel.Title = 'Machine Learning Setup';
            app.MachineLearningSetupPanel.FontWeight = 'bold';
            app.MachineLearningSetupPanel.FontSize = 16;
            app.MachineLearningSetupPanel.Position = [14 315 268 143];

            % Create ModelDropDownLabel
            app.ModelDropDownLabel = uilabel(app.MachineLearningSetupPanel);
            app.ModelDropDownLabel.HorizontalAlignment = 'center';
            app.ModelDropDownLabel.FontSize = 14;
            app.ModelDropDownLabel.Position = [105 93 45 22];
            app.ModelDropDownLabel.Text = 'Model';

            % Create MLModelDropDown
            app.MLModelDropDown = uidropdown(app.MachineLearningSetupPanel);
            app.MLModelDropDown.Items = {'None Trained'};
            app.MLModelDropDown.ValueChangedFcn = createCallbackFcn(app,
@MLModelDropDownValueChanged, true);
            app.MLModelDropDown.Position = [19 72 231 22];
            app.MLModelDropDown.Value = 'None Trained';

            % Create TrainedLabel
            app.TrainedLabel = uilabel(app.MachineLearningSetupPanel);
            app.TrainedLabel.FontSize = 14;
            app.TrainedLabel.Position = [19 48 55 22];
            app.TrainedLabel.Text = 'Trained:';

            % Create TypeLabel
            app.TypeLabel = uilabel(app.MachineLearningSetupPanel);
            app.TypeLabel.FontSize = 14;
            app.TypeLabel.Position = [19 25 39 22];
            app.TypeLabel.Text = 'Type:';

            % Create AccuracyLabel
            app.AccuracyLabel = uilabel(app.MachineLearningSetupPanel);
            app.AccuracyLabel.FontSize = 14;
            app.AccuracyLabel.Position = [20 2 68 22];
            app.AccuracyLabel.Text = 'Accuracy:';

            % Create TypeLabel_2
            app.TypeLabel_2 = uilabel(app.MachineLearningSetupPanel);
            app.TypeLabel_2.WordWrap = 'on';
            app.TypeLabel_2.FontSize = 14;
            app.TypeLabel_2.Position = [92 25 149 22];
            app.TypeLabel_2.Text = 'Type';

            % Create AccuracyLabel_2
            app.AccuracyLabel_2 = uilabel(app.MachineLearningSetupPanel);
            app.AccuracyLabel_2.WordWrap = 'on';
            app.AccuracyLabel_2.FontSize = 14;
```

```
            app.AccuracyLabel_2.Position = [91 2 149 22];
            app.AccuracyLabel_2.Text = 'Accuracy';

            % Create TrainedLabel_2
            app.TrainedLabel_2 = uilabel(app.MachineLearningSetupPanel);
            app.TrainedLabel_2.WordWrap = 'on';
            app.TrainedLabel_2.FontSize = 14;
            app.TrainedLabel_2.Position = [91 48 149 22];
            app.TrainedLabel_2.Text = 'Trained';

            % Create ResultsPanel
            app.ResultsPanel = uipanel(app.MainPassFailApplicationTab);
            app.ResultsPanel.TitlePosition = 'centertop';
            app.ResultsPanel.Title = 'Results:';
            app.ResultsPanel.FontWeight = 'bold';
            app.ResultsPanel.FontSize = 16;
            app.ResultsPanel.Position = [13 10 268 129];

            % Create PredictedBladeCondition
            app.PredictedBladeCondition = uilamp(app.ResultsPanel);
            app.PredictedBladeCondition.Position = [8 7 64 64];
            app.PredictedBladeCondition.Color = [1 1 1];

            % Create BladeConditionLabel
            app.BladeConditionLabel = uilabel(app.ResultsPanel);
            app.BladeConditionLabel.FontSize = 14;
            app.BladeConditionLabel.Position = [4 77 108 22];
            app.BladeConditionLabel.Text = 'Blade Condition:';

            % Create PredictedClass
            app.PredictedClass = uilabel(app.ResultsPanel);
            app.PredictedClass.FontSize = 14;
            app.PredictedClass.FontWeight = 'bold';
            app.PredictedClass.Position = [111 77 108 22];
            app.PredictedClass.Text = 'UD/D';

            % Create ProbabilityBladeCondition
            app.ProbabilityBladeCondition = uilabel(app.ResultsPanel);
            app.ProbabilityBladeCondition.HorizontalAlignment = 'center';
            app.ProbabilityBladeCondition.WordWrap = 'on';
            app.ProbabilityBladeCondition.FontSize = 14;
            app.ProbabilityBladeCondition.Position = [79 7 186 64];
            app.ProbabilityBladeCondition.Text = 'Probability it is undamaged';

            % Create TestSetupPanel
            app.TestSetupPanel = uipanel(app.MainPassFailApplicationTab);
            app.TestSetupPanel.TitlePosition = 'centertop';
            app.TestSetupPanel.Title = 'Test Setup';
            app.TestSetupPanel.FontWeight = 'bold';
            app.TestSetupPanel.FontSize = 16;
            app.TestSetupPanel.Position = [13 585 268 131];

            % Create TestTypeButtonGroup
            app.TestTypeButtonGroup = uibuttongroup(app.TestSetupPanel);
            app.TestTypeButtonGroup.SelectionChangedFcn = createCallbackFcn(app,
@TestTypeButtonGroupSelectionChanged, true);
            app.TestTypeButtonGroup.TitlePosition = 'centertop';
            app.TestTypeButtonGroup.Title = 'Test Type';
            app.TestTypeButtonGroup.FontSize = 14;
            app.TestTypeButtonGroup.Position = [0 57 268 50];

            % Create LiveButton
            app.LiveButton = uiradiobutton(app.TestTypeButtonGroup);
            app.LiveButton.Text = 'Live';
            app.LiveButton.FontSize = 14;
            app.LiveButton.Position = [4 2 47 22];
            app.LiveButton.Value = true;
```

```
            % Create SimulatedButton
            app.SimulatedButton = uiradiobutton(app.TestTypeButtonGroup);
            app.SimulatedButton.Text = 'Simulated';
            app.SimulatedButton.FontSize = 14;
            app.SimulatedButton.Position = [183 2 85 22];

            % Create BrowseButtonFile
            app.BrowseButtonFile = uibutton(app.TestSetupPanel, 'push');
            app.BrowseButtonFile.ButtonPushedFcn = createCallbackFcn(app,
@BrowseButtonFileMLPushed, true);
            app.BrowseButtonFile.FontSize = 14;
            app.BrowseButtonFile.Position = [3 4 74 49];
            app.BrowseButtonFile.Text = 'Browse';

            % Create TheSimulatedTestFileLabel
            app.TheSimulatedTestFileLabel = uilabel(app.TestSetupPanel);
            app.TheSimulatedTestFileLabel.FontSize = 14;
            app.TheSimulatedTestFileLabel.Position = [83 31 182 22];
            app.TheSimulatedTestFileLabel.Text = 'The Simulated Test File:';

            % Create SelectedTestFileLabel
            app.SelectedTestFileLabel = uilabel(app.TestSetupPanel);
            app.SelectedTestFileLabel.VerticalAlignment = 'top';
            app.SelectedTestFileLabel.WordWrap = 'on';
            app.SelectedTestFileLabel.FontSize = 14;
            app.SelectedTestFileLabel.Position = [83 8 182 19];
            app.SelectedTestFileLabel.Text = 'Selected Test File';

            % Create StartButtonMLTest
            app.StartButtonMLTest = uibutton(app.MainPassFailApplicationTab,
'push');
            app.StartButtonMLTest.ButtonPushedFcn = createCallbackFcn(app,
@StartButtonMLTestPushed, true);
            app.StartButtonMLTest.FontSize = 14;
            app.StartButtonMLTest.Position = [14 232 91 63];
            app.StartButtonMLTest.Text = 'Start';

            % Create PlotOptionsPanel
            app.PlotOptionsPanel = uipanel(app.MainPassFailApplicationTab);
            app.PlotOptionsPanel.TitlePosition = 'centertop';
            app.PlotOptionsPanel.Title = 'Plot Options';
            app.PlotOptionsPanel.FontWeight = 'bold';
            app.PlotOptionsPanel.FontSize = 16;
            app.PlotOptionsPanel.Position = [14 476 267 92];

            % Create HoldOnPlotsSwitchLabel
            app.HoldOnPlotsSwitchLabel = uilabel(app.PlotOptionsPanel);
            app.HoldOnPlotsSwitchLabel.HorizontalAlignment = 'center';
            app.HoldOnPlotsSwitchLabel.FontSize = 14;
            app.HoldOnPlotsSwitchLabel.Position = [34 35 93 22];
            app.HoldOnPlotsSwitchLabel.Text = 'Hold On Plots';

            % Create HoldOnPlotsSwitch
            app.HoldOnPlotsSwitch = uiswitch(app.PlotOptionsPanel, 'slider');
            app.HoldOnPlotsSwitch.Items = {'No', 'Yes'};
            app.HoldOnPlotsSwitch.FontSize = 14;
            app.HoldOnPlotsSwitch.Position = [174 38 40 17];
            app.HoldOnPlotsSwitch.Value = 'No';

            % Create FrequencyMaxkHzSpinnerLabel
            app.FrequencyMaxkHzSpinnerLabel = uilabel(app.PlotOptionsPanel);
            app.FrequencyMaxkHzSpinnerLabel.HorizontalAlignment = 'right';
            app.FrequencyMaxkHzSpinnerLabel.FontSize = 14;
            app.FrequencyMaxkHzSpinnerLabel.Position = [15 5 137 22];
            app.FrequencyMaxkHzSpinnerLabel.Text = 'Frequency Max (kHz)';

            % Create FrequencyMaxkHzSpinner
            app.FrequencyMaxkHzSpinner = uispinner(app.PlotOptionsPanel);
```

```
            app.FrequencyMaxkHzSpinner.Limits = [0 Inf];
            app.FrequencyMaxkHzSpinner.Position = [167 5 88 22];
            app.FrequencyMaxkHzSpinner.Value = 50;

            % Create StopButtonMLTest
            app.StopButtonMLTest = uibutton(app.MainPassFailApplicationTab,
'push');
            app.StopButtonMLTest.ButtonPushedFcn = createCallbackFcn(app,
@StopButtonPushed, true);
            app.StopButtonMLTest.BackgroundColor = [1 0 0];
            app.StopButtonMLTest.FontSize = 14;
            app.StopButtonMLTest.Position = [190 232 91 63];
            app.StopButtonMLTest.Text = 'Stop';

            % Create ProgressMLTest
            app.ProgressMLTest = uibutton(app.MainPassFailApplicationTab, 'push');
            app.ProgressMLTest.FontSize = 14;
            app.ProgressMLTest.Position = [14 153 267 63];
            app.ProgressMLTest.Text = 'Machine Learning Test Progress';

            % Create UIAxesRaw_Main
            app.UIAxesRaw_Main = uiaxes(app.MainPassFailApplicationTab);
            title(app.UIAxesRaw_Main, 'Raw Signal Data')
            xlabel(app.UIAxesRaw_Main, 'Time (ms)')
            ylabel(app.UIAxesRaw_Main, 'Signal Amplitude (V)')
            zlabel(app.UIAxesRaw_Main, 'Z')
            app.UIAxesRaw_Main.PlotBoxAspectRatio = [3.06737588652482 1 1];
            app.UIAxesRaw_Main.FontSize = 14;
            app.UIAxesRaw_Main.Position = [293 369 922 347];

            % Create UIAxesFFTAmp
            app.UIAxesFFTAmp = uiaxes(app.MainPassFailApplicationTab);
            title(app.UIAxesFFTAmp, 'Power Spectrum Amplitude')
            xlabel(app.UIAxesFFTAmp, 'Frequency (kHz)')
            ylabel(app.UIAxesFFTAmp, 'Amplitude')
            zlabel(app.UIAxesFFTAmp, 'Z')
            app.UIAxesFFTAmp.PlotBoxAspectRatio = [3.10035842293907 1 1];
            app.UIAxesFFTAmp.FontSize = 14;
            app.UIAxesFFTAmp.Position = [293 10 922 344];

            % Create CollectMoreTrainingDataTab
            app.CollectMoreTrainingDataTab = uitab(app.TabGroup);
            app.CollectMoreTrainingDataTab.Title = 'Collect More Training Data';

            % Create TestSetupPanel_2
            app.TestSetupPanel_2 = uipanel(app.CollectMoreTrainingDataTab);
            app.TestSetupPanel_2.TitlePosition = 'centertop';
            app.TestSetupPanel_2.Title = 'Test Setup';
            app.TestSetupPanel_2.FontWeight = 'bold';
            app.TestSetupPanel_2.FontSize = 16;
            app.TestSetupPanel_2.Position = [13 623 1205 93];

            % Create BrowseButtonFolderTestData
            app.BrowseButtonFolderTestData = uibutton(app.TestSetupPanel_2,
'push');
            app.BrowseButtonFolderTestData.ButtonPushedFcn = createCallbackFcn(app,
@BrowseButtonFolderTestDataPushed, true);
            app.BrowseButtonFolderTestData.FontSize = 14;
            app.BrowseButtonFolderTestData.Position = [5 6 77 49];
            app.BrowseButtonFolderTestData.Text = 'Browse';

            % Create TheBladesTopLevelFolderPathLabel_3
            app.TheBladesTopLevelFolderPathLabel_3 = uilabel(app.TestSetupPanel_2);
            app.TheBladesTopLevelFolderPathLabel_3.FontSize = 14;
            app.TheBladesTopLevelFolderPathLabel_3.Position = [87 43 220 22];
            app.TheBladesTopLevelFolderPathLabel_3.Text = 'The Blades Top Level
Folder Path:';
```

```
            % Create SelectedFolderPathLabel
            app.SelectedFolderPathLabel = uilabel(app.TestSetupPanel_2);
            app.SelectedFolderPathLabel.WordWrap = 'on';
            app.SelectedFolderPathLabel.FontSize = 14;
            app.SelectedFolderPathLabel.Position = [87 5 370 40];
            app.SelectedFolderPathLabel.Text = 'Selected Folder Path';

            % Create FileNameEditFieldLabel
            app.FileNameEditFieldLabel = uilabel(app.TestSetupPanel_2);
            app.FileNameEditFieldLabel.HorizontalAlignment = 'center';
            app.FileNameEditFieldLabel.FontSize = 14;
            app.FileNameEditFieldLabel.Position = [474 17 68 22];
            app.FileNameEditFieldLabel.Text = 'File Name';

            % Create FileNameEditField
            app.FileNameEditField = uieditfield(app.TestSetupPanel_2, 'text');
            app.FileNameEditField.Position = [552 17 177 22];
            app.FileNameEditField.Value = 'SN12345_UD';

            % Create NumberofTestsSpinnerLabel
            app.NumberofTestsSpinnerLabel = uilabel(app.TestSetupPanel_2);
            app.NumberofTestsSpinnerLabel.HorizontalAlignment = 'right';
            app.NumberofTestsSpinnerLabel.Position = [966 17 94 22];
            app.NumberofTestsSpinnerLabel.Text = 'Number of Tests';

            % Create NumberofTestsSpinner
            app.NumberofTestsSpinner = uispinner(app.TestSetupPanel_2);
            app.NumberofTestsSpinner.Limits = [1 Inf];
            app.NumberofTestsSpinner.Position = [1075 17 73 22];
            app.NumberofTestsSpinner.Value = 1;

            % Create StartButtonTestData
            app.StartButtonTestData = uibutton(app.CollectMoreTrainingDataTab,
'push');
            app.StartButtonTestData.ButtonPushedFcn = createCallbackFcn(app,
@StartButtonTestDataPushed, true);
            app.StartButtonTestData.FontSize = 14;
            app.StartButtonTestData.Position = [487 531 91 63];
            app.StartButtonTestData.Text = 'Start';

            % Create StopButtonTestData
            app.StopButtonTestData = uibutton(app.CollectMoreTrainingDataTab,
'push');
            app.StopButtonTestData.ButtonPushedFcn = createCallbackFcn(app,
@StopButtonPushed, true);
            app.StopButtonTestData.BackgroundColor = [1 0 0];
            app.StopButtonTestData.FontSize = 14;
            app.StopButtonTestData.Position = [651 531 91 63];
            app.StopButtonTestData.Text = 'Stop';

            % Create ProgressTestData
            app.ProgressTestData = uibutton(app.CollectMoreTrainingDataTab,
'push');
            app.ProgressTestData.FontSize = 14;
            app.ProgressTestData.Position = [402 391 427 95];
            app.ProgressTestData.Text = 'Training Data Collection Progress';

            % Create UIAxesRaw_Col
            app.UIAxesRaw_Col = uiaxes(app.CollectMoreTrainingDataTab);
            title(app.UIAxesRaw_Col, 'Raw Signal Data')
            xlabel(app.UIAxesRaw_Col, 'Time (ms)')
            ylabel(app.UIAxesRaw_Col, 'Signal Amplitude (V)')
            zlabel(app.UIAxesRaw_Col, 'Z')
            app.UIAxesRaw_Col.PlotBoxAspectRatio = [3.88813559322034 1 1];
            app.UIAxesRaw_Col.FontSize = 14;
            app.UIAxesRaw_Col.Position = [14 10 1204 360];

            % Create ManualFeatureExtractionTab
```

```
            app.ManualFeatureExtractionTab = uitab(app.TabGroup);
            app.ManualFeatureExtractionTab.Title = 'Manual Feature Extraction';

            % Create StartButtonFeatExtract
            app.StartButtonFeatExtract = uibutton(app.ManualFeatureExtractionTab,
'push');
            app.StartButtonFeatExtract.ButtonPushedFcn = createCallbackFcn(app,
@StartButtonReTrainPeaksPushed, true);
            app.StartButtonFeatExtract.FontSize = 14;
            app.StartButtonFeatExtract.Position = [14 538 149 63];
            app.StartButtonFeatExtract.Text = 'Begin Peak Selection';

            % Create StopButtonFeatExtract
            app.StopButtonFeatExtract = uibutton(app.ManualFeatureExtractionTab,
'push');
            app.StopButtonFeatExtract.ButtonPushedFcn = createCallbackFcn(app,
@StopButtonPushed, true);
            app.StopButtonFeatExtract.BackgroundColor = [1 0 0];
            app.StopButtonFeatExtract.FontSize = 14;
            app.StopButtonFeatExtract.Position = [251 538 149 63];
            app.StopButtonFeatExtract.Text = 'Stop Peak Selection';

            % Create NextFileButtonFeatExtract
            app.NextFileButtonFeatExtract =
uibutton(app.ManualFeatureExtractionTab, 'push');
            app.NextFileButtonFeatExtract.ButtonPushedFcn = createCallbackFcn(app,
@NextFileButtonPushed, true);
            app.NextFileButtonFeatExtract.FontSize = 14;
            app.NextFileButtonFeatExtract.Position = [487 538 149 63];
            app.NextFileButtonFeatExtract.Text = 'Try Next File';

            % Create RandomFileLabelFeatExtract
            app.RandomFileLabelFeatExtract =
uilabel(app.ManualFeatureExtractionTab);
            app.RandomFileLabelFeatExtract.FontSize = 14;
            app.RandomFileLabelFeatExtract.Position = [735 538 88 22];
            app.RandomFileLabelFeatExtract.Text = 'Random File:';

            % Create SelectedFeatExtract
            app.SelectedFeatExtract = uilabel(app.ManualFeatureExtractionTab);
            app.SelectedFeatExtract.WordWrap = 'on';
            app.SelectedFeatExtract.FontSize = 14;
            app.SelectedFeatExtract.Position = [827 539 163 20];
            app.SelectedFeatExtract.Text = 'Selected File';

            % Create DeleteRowButtonFeatExtract
            app.DeleteRowButtonFeatExtract =
uibutton(app.ManualFeatureExtractionTab, 'push');
            app.DeleteRowButtonFeatExtract.ButtonPushedFcn = createCallbackFcn(app,
@DeleteRowButtonFeatExtractPushed, true);
            app.DeleteRowButtonFeatExtract.FontSize = 14;
            app.DeleteRowButtonFeatExtract.Position = [1113 528 64 39];
            app.DeleteRowButtonFeatExtract.Text = 'Delete';

            % Create ManualFeatureExtractionSetupPanel
            app.ManualFeatureExtractionSetupPanel =
uipanel(app.ManualFeatureExtractionTab);
            app.ManualFeatureExtractionSetupPanel.TitlePosition = 'centertop';
            app.ManualFeatureExtractionSetupPanel.Title = 'Manual Feature
Extraction Setup';
            app.ManualFeatureExtractionSetupPanel.FontWeight = 'bold';
            app.ManualFeatureExtractionSetupPanel.FontSize = 16;
            app.ManualFeatureExtractionSetupPanel.Position = [13 623 1204 93];

            % Create BrowseButtonFolderReML
            app.BrowseButtonFolderReML =
uibutton(app.ManualFeatureExtractionSetupPanel, 'push');
```

```
            app.BrowseButtonFolderReML.ButtonPushedFcn = createCallbackFcn(app,
@BrowseButtonFolderFeatExtractPushed, true);
            app.BrowseButtonFolderReML.FontSize = 14;
            app.BrowseButtonFolderReML.Position = [5 6 77 49];
            app.BrowseButtonFolderReML.Text = 'Browse';

            % Create TheBladesTopLevelFolderPathLabel
            app.TheBladesTopLevelFolderPathLabel =
uilabel(app.ManualFeatureExtractionSetupPanel);
            app.TheBladesTopLevelFolderPathLabel.FontSize = 14;
            app.TheBladesTopLevelFolderPathLabel.Position = [87 43 291 22];
            app.TheBladesTopLevelFolderPathLabel.Text = 'The Blades Top Level
Folder Path:';

            % Create SelectedFolderPathLabelFeatExtract
            app.SelectedFolderPathLabelFeatExtract =
uilabel(app.ManualFeatureExtractionSetupPanel);
            app.SelectedFolderPathLabelFeatExtract.WordWrap = 'on';
            app.SelectedFolderPathLabelFeatExtract.FontSize = 14;
            app.SelectedFolderPathLabelFeatExtract.Position = [87 6 344 38];
            app.SelectedFolderPathLabelFeatExtract.Text = 'Selected Folder Path';

            % Create ExtractedfeaturesmatchthemachinelearningalgorithmLabel
            app.ExtractedfeaturesmatchthemachinelearningalgorithmLabel =
uilabel(app.ManualFeatureExtractionSetupPanel);

app.ExtractedfeaturesmatchthemachinelearningalgorithmLabel.HorizontalAlignment =
'center';
            app.ExtractedfeaturesmatchthemachinelearningalgorithmLabel.WordWrap =
'on';
            app.ExtractedfeaturesmatchthemachinelearningalgorithmLabel.FontSize =
14;
            app.ExtractedfeaturesmatchthemachinelearningalgorithmLabel.Position =
[903 1 277 65];
            app.ExtractedfeaturesmatchthemachinelearningalgorithmLabel.Text =
'Manual feature extraction is ONLY required if the machine learning method being
used, is not deep learning';

            % Create FrequencyMaxkHzSpinner_2Label
            app.FrequencyMaxkHzSpinner_2Label =
uilabel(app.ManualFeatureExtractionSetupPanel);
            app.FrequencyMaxkHzSpinner_2Label.HorizontalAlignment = 'right';
            app.FrequencyMaxkHzSpinner_2Label.FontSize = 14;
            app.FrequencyMaxkHzSpinner_2Label.Position = [622 38 137 22];
            app.FrequencyMaxkHzSpinner_2Label.Text = 'Frequency Max (kHz)';

            % Create FrequencyMaxkHzSpinnerFeatExtract
            app.FrequencyMaxkHzSpinnerFeatExtract =
uispinner(app.ManualFeatureExtractionSetupPanel);
            app.FrequencyMaxkHzSpinnerFeatExtract.Limits = [0 Inf];
            app.FrequencyMaxkHzSpinnerFeatExtract.ValueChangedFcn =
createCallbackFcn(app, @FrequencyMaxkHzSpinnerFeatExtractValueChanged, true);
            app.FrequencyMaxkHzSpinnerFeatExtract.Position = [647 9 88 22];
            app.FrequencyMaxkHzSpinnerFeatExtract.Value = 50;

            % Create FrequencyMinkHzLabel
            app.FrequencyMinkHzLabel =
uilabel(app.ManualFeatureExtractionSetupPanel);
            app.FrequencyMinkHzLabel.HorizontalAlignment = 'right';
            app.FrequencyMinkHzLabel.FontSize = 14;
            app.FrequencyMinkHzLabel.Position = [430 38 134 22];
            app.FrequencyMinkHzLabel.Text = 'Frequency Min (kHz)';

            % Create FrequencyMinkHzSpinnerFeatExtract
            app.FrequencyMinkHzSpinnerFeatExtract =
uispinner(app.ManualFeatureExtractionSetupPanel);
            app.FrequencyMinkHzSpinnerFeatExtract.Limits = [0 Inf];
```

```
            app.FrequencyMinkHzSpinnerFeatExtract.ValueChangedFcn =
createCallbackFcn(app, @FrequencyMaxkHzSpinnerFeatExtractValueChanged, true);
            app.FrequencyMinkHzSpinnerFeatExtract.Position = [452 9 88 22];

            % Create UITablesFrequencyModes
            app.UITablesFrequencyModes = uitable(app.ManualFeatureExtractionTab);
            app.UITablesFrequencyModes.ColumnName = {'Mode'; 'Frequency (kHz)'};
            app.UITablesFrequencyModes.ColumnWidth = {45, 108};
            app.UITablesFrequencyModes.RowName = {};
            app.UITablesFrequencyModes.ColumnEditable = [false false];
            app.UITablesFrequencyModes.FontSize = 14;
            app.UITablesFrequencyModes.Position = [1067 52 155 442];

            % Create RownumberEditFieldLabel
            app.RownumberEditFieldLabel = uilabel(app.ManualFeatureExtractionTab);
            app.RownumberEditFieldLabel.HorizontalAlignment = 'right';
            app.RownumberEditFieldLabel.FontSize = 14;
            app.RownumberEditFieldLabel.Position = [1082 574 86 22];
            app.RownumberEditFieldLabel.Text = 'Row number';

            % Create RowNumberEditFieldFeatExtract
            app.RowNumberEditFieldFeatExtract =
uieditfield(app.ManualFeatureExtractionTab, 'numeric');
            app.RowNumberEditFieldFeatExtract.Limits = [1 Inf];
            app.RowNumberEditFieldFeatExtract.Position = [1179 574 26 22];
            app.RowNumberEditFieldFeatExtract.Value = 1;

            % Create ConditionLabelFeatExtract
            app.ConditionLabelFeatExtract =
uilabel(app.ManualFeatureExtractionTab);
            app.ConditionLabelFeatExtract.HorizontalAlignment = 'center';
            app.ConditionLabelFeatExtract.FontSize = 14;
            app.ConditionLabelFeatExtract.Position = [735 574 70 22];
            app.ConditionLabelFeatExtract.Text = 'Condition:';

            % Create BladeConditionSwitchFeatExtract
            app.BladeConditionSwitchFeatExtract =
uiswitch(app.ManualFeatureExtractionTab, 'slider');
            app.BladeConditionSwitchFeatExtract.Items = {'Undamaged', 'Damaged'};
            app.BladeConditionSwitchFeatExtract.ValueChangedFcn =
createCallbackFcn(app, @NextFileButtonPushed, true);
            app.BladeConditionSwitchFeatExtract.FontSize = 14;
            app.BladeConditionSwitchFeatExtract.Position = [904 577 40 17];
            app.BladeConditionSwitchFeatExtract.Value = 'Undamaged';

            % Create UIAxesFeatExtract
            app.UIAxesFeatExtract = uiaxes(app.ManualFeatureExtractionTab);
            title(app.UIAxesFeatExtract, 'Power Spectrum Amplitude')
            xlabel(app.UIAxesFeatExtract, 'Frequency (kHz)')
            ylabel(app.UIAxesFeatExtract, 'Amplitude')
            zlabel(app.UIAxesFeatExtract, 'Z')
            app.UIAxesFeatExtract.PlotBoxAspectRatio = [2.26410835214447 1 1];
            app.UIAxesFeatExtract.FontSize = 14;
            app.UIAxesFeatExtract.Position = [13 10 1060 508];

            % Create RetrainMachineLearningModelTab
            app.RetrainMachineLearningModelTab = uitab(app.TabGroup);
            app.RetrainMachineLearningModelTab.Title = 'Retrain Machine Learning
Model';

            % Create RetrainMachineLearningModelSetupPanel
            app.RetrainMachineLearningModelSetupPanel =
uipanel(app.RetrainMachineLearningModelTab);
            app.RetrainMachineLearningModelSetupPanel.TitlePosition = 'centertop';
            app.RetrainMachineLearningModelSetupPanel.Title = 'Retrain Machine
Learning Model Setup';
            app.RetrainMachineLearningModelSetupPanel.FontWeight = 'bold';
            app.RetrainMachineLearningModelSetupPanel.FontSize = 16;
```

```
            app.RetrainMachineLearningModelSetupPanel.Position = [13 623 1204 93];

            % Create BrowseButtonFolderReTrainML
            app.BrowseButtonFolderReTrainML =
uibutton(app.RetrainMachineLearningModelSetupPanel, 'push');
            app.BrowseButtonFolderReTrainML.ButtonPushedFcn =
createCallbackFcn(app, @BrowseButtonFolderReTrainMLPushed, true);
            app.BrowseButtonFolderReTrainML.FontSize = 14;
            app.BrowseButtonFolderReTrainML.Position = [5 6 77 49];
            app.BrowseButtonFolderReTrainML.Text = 'Browse';

            % Create TheBladesTopLevelFolderPathLabel_2
            app.TheBladesTopLevelFolderPathLabel_2 =
uilabel(app.RetrainMachineLearningModelSetupPanel);
            app.TheBladesTopLevelFolderPathLabel_2.FontSize = 14;
            app.TheBladesTopLevelFolderPathLabel_2.Position = [87 43 291 22];
            app.TheBladesTopLevelFolderPathLabel_2.Text = 'The Blades Top Level
Folder Path:';

            % Create SelectedFolderPathLabelReTrainML
            app.SelectedFolderPathLabelReTrainML =
uilabel(app.RetrainMachineLearningModelSetupPanel);
            app.SelectedFolderPathLabelReTrainML.WordWrap = 'on';
            app.SelectedFolderPathLabelReTrainML.FontSize = 14;
            app.SelectedFolderPathLabelReTrainML.Position = [87 6 330 38];
            app.SelectedFolderPathLabelReTrainML.Text = 'Selected Folder Path';

            % Create TrainingDataAvailableButtonGroup
            app.TrainingDataAvailableButtonGroup =
uibuttongroup(app.RetrainMachineLearningModelSetupPanel);
            app.TrainingDataAvailableButtonGroup.SelectionChangedFcn =
createCallbackFcn(app, @TrainingDataAvailableButtonGroupSelectionChanged, true);
            app.TrainingDataAvailableButtonGroup.TitlePosition = 'centertop';
            app.TrainingDataAvailableButtonGroup.Title = 'Training Data
Available?';
            app.TrainingDataAvailableButtonGroup.FontSize = 14;
            app.TrainingDataAvailableButtonGroup.Position = [438 0 327 69];

            % Create UndamagedandDamagedButton
            app.UndamagedandDamagedButton =
uiradiobutton(app.TrainingDataAvailableButtonGroup);
            app.UndamagedandDamagedButton.Text = 'Undamaged and Damaged';
            app.UndamagedandDamagedButton.FontSize = 14;
            app.UndamagedandDamagedButton.Position = [5 14 192 22];
            app.UndamagedandDamagedButton.Value = true;

            % Create UndamagedButton
            app.UndamagedButton =
uiradiobutton(app.TrainingDataAvailableButtonGroup);
            app.UndamagedButton.Text = 'Undamaged';
            app.UndamagedButton.FontSize = 14;
            app.UndamagedButton.Position = [228 14 99 22];

            % Create NumberOfFeaturesText
            app.NumberOfFeaturesText =
uilabel(app.RetrainMachineLearningModelSetupPanel);
            app.NumberOfFeaturesText.FontSize = 14;
            app.NumberOfFeaturesText.Position = [847 21 299 22];
            app.NumberOfFeaturesText.Text = {'Number of extracted features to use
in training:'; ''};

            % Create NumberOfFeatures
            app.NumberOfFeatures =
uilabel(app.RetrainMachineLearningModelSetupPanel);
            app.NumberOfFeatures.FontSize = 14;
            app.NumberOfFeatures.FontWeight = 'bold';
            app.NumberOfFeatures.Position = [1147 21 25 22];
            app.NumberOfFeatures.Text = 'NA';
```

```
            % Create MachineLearningMethodtoTrainListBoxLabel
            app.MachineLearningMethodtoTrainListBoxLabel =
uilabel(app.RetrainMachineLearningModelTab);
            app.MachineLearningMethodtoTrainListBoxLabel.HorizontalAlignment =
'center';
            app.MachineLearningMethodtoTrainListBoxLabel.FontSize = 14;
            app.MachineLearningMethodtoTrainListBoxLabel.Position = [493 559 220
22];
            app.MachineLearningMethodtoTrainListBoxLabel.Text = 'Machine Learning
Method to Train';

            % Create MachineLearningMethodtoTrainListBox
            app.MachineLearningMethodtoTrainListBox =
uilistbox(app.RetrainMachineLearningModelTab);
            app.MachineLearningMethodtoTrainListBox.Items = {'Algorithms1',
'Algorithms2', 'Algorithms3', 'Algorithms4', 'Algorithms5'};
            app.MachineLearningMethodtoTrainListBox.FontSize = 14;
            app.MachineLearningMethodtoTrainListBox.Position = [538 411 130 145];
            app.MachineLearningMethodtoTrainListBox.Value = 'Algorithms1';

            % Create MLmodelTable
            app.MLmodelTable = uitable(app.RetrainMachineLearningModelTab);
            app.MLmodelTable.ColumnName = {'Model'; 'Trained'; 'Type'; 'Accuracy'};
            app.MLmodelTable.ColumnWidth = {'auto', 100, 70, 70};
            app.MLmodelTable.RowName = {};
            app.MLmodelTable.FontSize = 14;
            app.MLmodelTable.Position = [14 206 464 375];

            % Create ConfusionChartPanel
            app.ConfusionChartPanel = uipanel(app.RetrainMachineLearningModelTab);
            app.ConfusionChartPanel.TitlePosition = 'centertop';
            app.ConfusionChartPanel.Title = 'Confusion Chart';
            app.ConfusionChartPanel.FontSize = 14;
            app.ConfusionChartPanel.Position = [725 153 490 428];

            % Create StartButtonReTrainML
            app.StartButtonReTrainML = uibutton(app.RetrainMachineLearningModelTab,
'push');
            app.StartButtonReTrainML.ButtonPushedFcn = createCallbackFcn(app,
@StartButtonReTrainMLPushed, true);
            app.StartButtonReTrainML.FontSize = 14;
            app.StartButtonReTrainML.Position = [539 232 133 63];
            app.StartButtonReTrainML.Text = 'Begin Training';

            % Create CurrentTrainedModelsLabel_2
            app.CurrentTrainedModelsLabel_2 =
uilabel(app.RetrainMachineLearningModelTab);
            app.CurrentTrainedModelsLabel_2.VerticalAlignment = 'top';
            app.CurrentTrainedModelsLabel_2.WordWrap = 'on';
            app.CurrentTrainedModelsLabel_2.FontSize = 14;
            app.CurrentTrainedModelsLabel_2.Position = [926 18 289 112];
            app.CurrentTrainedModelsLabel_2.Text = {'The speed of training will be
drastically impacted by the hardware the application is running on. A computer with
an external GPU will train the models much quicker. '; 'The DL and AUTO models will
take the longest time to train, whilst the others should be completed in a matter
of minutues.'; ''};

            % Create AccuracyofthemodelLabel
            app.AccuracyofthemodelLabel =
uilabel(app.RetrainMachineLearningModelTab);
            app.AccuracyofthemodelLabel.FontSize = 14;
            app.AccuracyofthemodelLabel.Position = [482 185 150 22];
            app.AccuracyofthemodelLabel.Text = 'Accuracy of the model:';

            % Create AccuracyOfTheModel
            app.AccuracyOfTheModel = uilabel(app.RetrainMachineLearningModelTab);
            app.AccuracyOfTheModel.FontSize = 14;
```

```
            app.AccuracyOfTheModel.FontWeight = 'bold';
            app.AccuracyOfTheModel.Position = [633 185 104 22];
            app.AccuracyOfTheModel.Text = 'NA';

            % Create LossofthemodelLabel
            app.LossofthemodelLabel = uilabel(app.RetrainMachineLearningModelTab);
            app.LossofthemodelLabel.FontSize = 14;
            app.LossofthemodelLabel.Position = [482 155 151 22];
            app.LossofthemodelLabel.Text = 'Loss of the model:';

            % Create LossOfTheModel
            app.LossOfTheModel = uilabel(app.RetrainMachineLearningModelTab);
            app.LossOfTheModel.FontSize = 14;
            app.LossOfTheModel.FontWeight = 'bold';
            app.LossOfTheModel.Position = [633 155 104 22];
            app.LossOfTheModel.Text = 'NA';

            % Create CurrentTrainedModelsLabel_3
            app.CurrentTrainedModelsLabel_3 =
uilabel(app.RetrainMachineLearningModelTab);
            app.CurrentTrainedModelsLabel_3.VerticalAlignment = 'top';
            app.CurrentTrainedModelsLabel_3.WordWrap = 'on';
            app.CurrentTrainedModelsLabel_3.FontSize = 14;
            app.CurrentTrainedModelsLabel_3.Position = [13 17 229 112];
            app.CurrentTrainedModelsLabel_3.Text = {'AUTO - Selects the best
method'; 'Tree - Binary classification tree'; 'kNN - k-nearest neighbours'; 'NB -
Naive Bayes'; 'DA - Discriminant analysis'; 'ShallowNN - Shallow neural network';
'DL - Deep learning'; ''};

            % Create CurrentTrainedModelsLabel_4
            app.CurrentTrainedModelsLabel_4 =
uilabel(app.RetrainMachineLearningModelTab);
            app.CurrentTrainedModelsLabel_4.VerticalAlignment = 'top';
            app.CurrentTrainedModelsLabel_4.WordWrap = 'on';
            app.CurrentTrainedModelsLabel_4.FontSize = 14;
            app.CurrentTrainedModelsLabel_4.Position = [256 2 297 128];
            app.CurrentTrainedModelsLabel_4.Text = {'Basic - Simple std statistical
analysis'; 'SpectralCluster - graph-based algorithm'; 'Gaussian - Gaussian mixture
model'; 'DBSCAN - Density-based spatial clustering of applications with noise';
'(Auto reduce features is not required because the training time is already very
quick)'; ''};

            % Create AutoReduceFeaturesButtonGroup
            app.AutoReduceFeaturesButtonGroup =
uibuttongroup(app.RetrainMachineLearningModelTab);
            app.AutoReduceFeaturesButtonGroup.TitlePosition = 'centertop';
            app.AutoReduceFeaturesButtonGroup.Title = 'Auto Reduce Features?';
            app.AutoReduceFeaturesButtonGroup.FontSize = 14;
            app.AutoReduceFeaturesButtonGroup.Position = [521 353 165 49];

            % Create YesButton
            app.YesButton = uiradiobutton(app.AutoReduceFeaturesButtonGroup);
            app.YesButton.Text = 'Yes';
            app.YesButton.FontSize = 14;
            app.YesButton.Position = [1 3 44 22];

            % Create NoButton
            app.NoButton = uiradiobutton(app.AutoReduceFeaturesButtonGroup);
            app.NoButton.Text = 'No';
            app.NoButton.FontSize = 14;
            app.NoButton.Position = [125 3 40 22];
            app.NoButton.Value = true;

            % Create Descp
            app.Descp = uilabel(app.RetrainMachineLearningModelTab);
            app.Descp.WordWrap = 'on';
            app.Descp.FontSize = 14;
            app.Descp.Position = [522 318 165 36];
```

```
            app.Descp.Text = '(May reduce accuracy but will speed up training)';

            % Create CurrentTrainedModelsLabel_5
            app.CurrentTrainedModelsLabel_5 =
uilabel(app.RetrainMachineLearningModelTab);
            app.CurrentTrainedModelsLabel_5.VerticalAlignment = 'top';
            app.CurrentTrainedModelsLabel_5.WordWrap = 'on';
            app.CurrentTrainedModelsLabel_5.FontSize = 14;
            app.CurrentTrainedModelsLabel_5.Position = [573 18 255 112];
            app.CurrentTrainedModelsLabel_5.Text = {'For undamaged and damaged data
the best starting methods is kNN'; 'For undamaged the best starting method is
DBSCAN'; ''};

            % Create DeleteRowButtonReTrainML
            app.DeleteRowButtonReTrainML =
uibutton(app.RetrainMachineLearningModelTab, 'push');
            app.DeleteRowButtonReTrainML.ButtonPushedFcn = createCallbackFcn(app,
@DeleteRowButtonReTrainMLPushed, true);
            app.DeleteRowButtonReTrainML.FontSize = 14;
            app.DeleteRowButtonReTrainML.Position = [327 157 73 43];
            app.DeleteRowButtonReTrainML.Text = 'Delete';

            % Create ModeltoDeleteDropDownLabel
            app.ModeltoDeleteDropDownLabel =
uilabel(app.RetrainMachineLearningModelTab);
            app.ModeltoDeleteDropDownLabel.HorizontalAlignment = 'center';
            app.ModeltoDeleteDropDownLabel.FontSize = 14;
            app.ModeltoDeleteDropDownLabel.Position = [119 178 105 22];
            app.ModeltoDeleteDropDownLabel.Text = 'Model to Delete';

            % Create ModeltoDeleteDropDown
            app.ModeltoDeleteDropDown =
uidropdown(app.RetrainMachineLearningModelTab);
            app.ModeltoDeleteDropDown.Items = {'None Trained'};
            app.ModeltoDeleteDropDown.Position = [63 157 231 22];
            app.ModeltoDeleteDropDown.Value = 'None Trained';

            % Create StatusLabel
            app.StatusLabel = uilabel(app.UIFigure);
            app.StatusLabel.BackgroundColor = [1 1 1];
            app.StatusLabel.FontSize = 14;
            app.StatusLabel.Position = [46 6 1173 21];
            app.StatusLabel.Text = 'Status';

            % Create ErrorLamp
            app.ErrorLamp = uilamp(app.UIFigure);
            app.ErrorLamp.Position = [4 3 26 26];

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Pass_Fail_App_exported

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)
```

```
            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Appendix M – User Manual

Software and Drivers to Install

- The MATLAB Pass/Fail application with sub-functions must be downloaded and navigated to the root directory of MATLAB. The code can be downloaded from GitHub. All the subfolders downloaded from GitHub also need to be added to the MATLAB path.

- The drivers for the PS4000 PicoScope must also be downloaded which have been tested on Windows, they should work on Mac but are very temperamental; contact PicoScope for support if usage on mac is essential and mention the SUPPORT-26796 ticket. These drivers be found on the PicoScope website here, navigate to 'Discontinued Series', 'PicoScope 4424' and download:
    - PicoSDK for your system
    - PicoScope 6 for your system

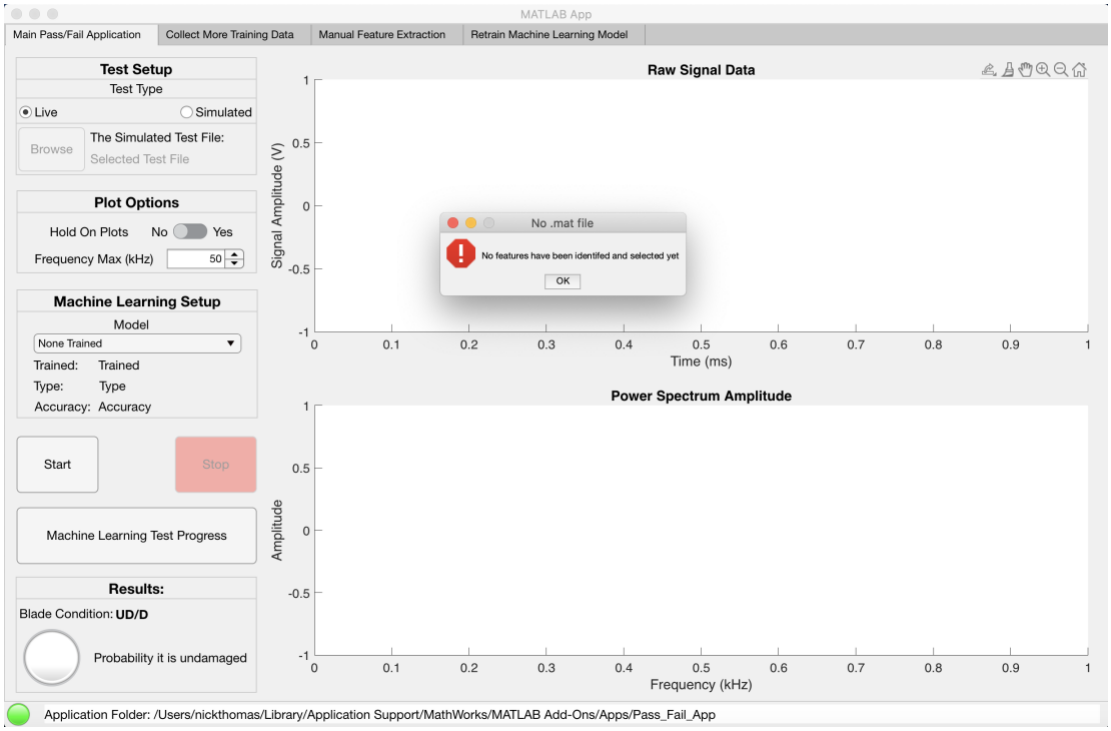
- There are some extra MATLAB toolboxes which are required to run the application as well as communicating with the hardware. These can be added from the 'Add-Ons' button in MATLAB and are:
    - Instrument Control Toolbox
    - Parallel Computing Toolbox
    - PicoScope Support Toolbox
    - PicoScope 4000 Series MATLAB Generic Instrument Driver
    - Signal Processing Toolbox
    - Statistics and Machine Learning Toolbox

The file structure of the downloaded MATLAB code should be preserved. The important aspect is to ensure wherever the training data is kept, that it has two subfolders, 'Damaged' and 'Undamaged'; these are case sensitive!

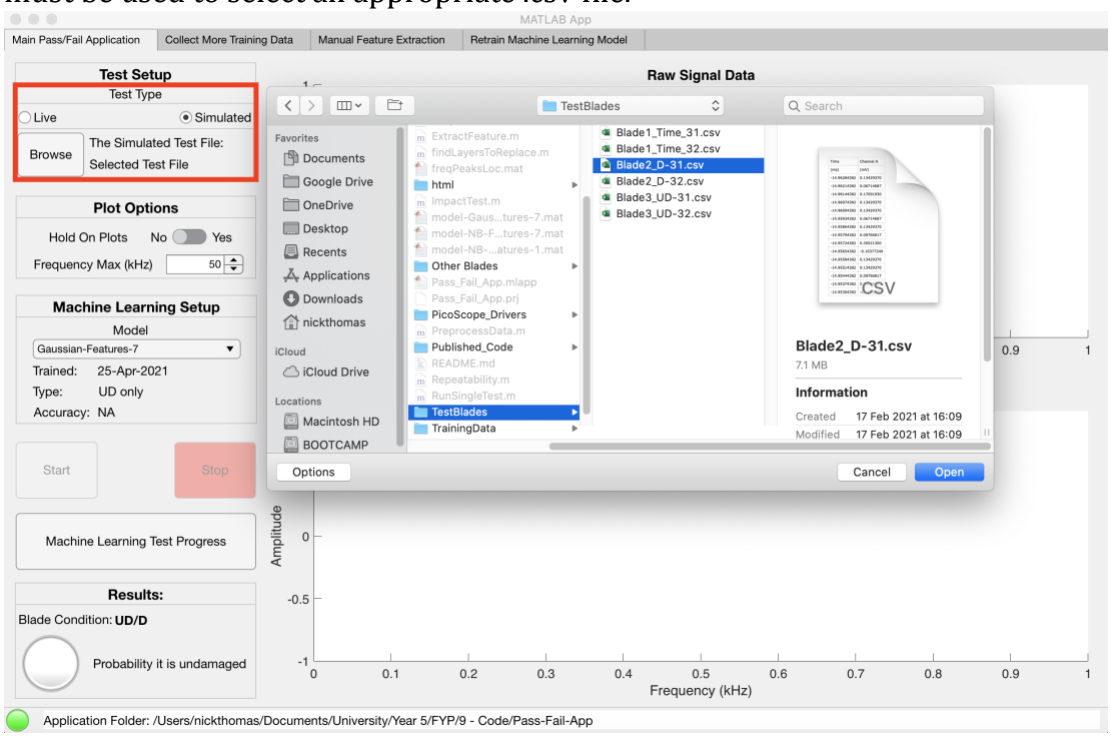
To run the application, simply open the 'Pass_Fail_App.mlapp' from the downloaded folders and click on the *Run* button. The alternative is to run it directly from the 'APPS' tab in MATLAB, if the application has been installed, by clicking the 'Pass_Fail_App_mlappinstall' file from the downloaded folder.

Main Pass/Fail Application Tab Guide

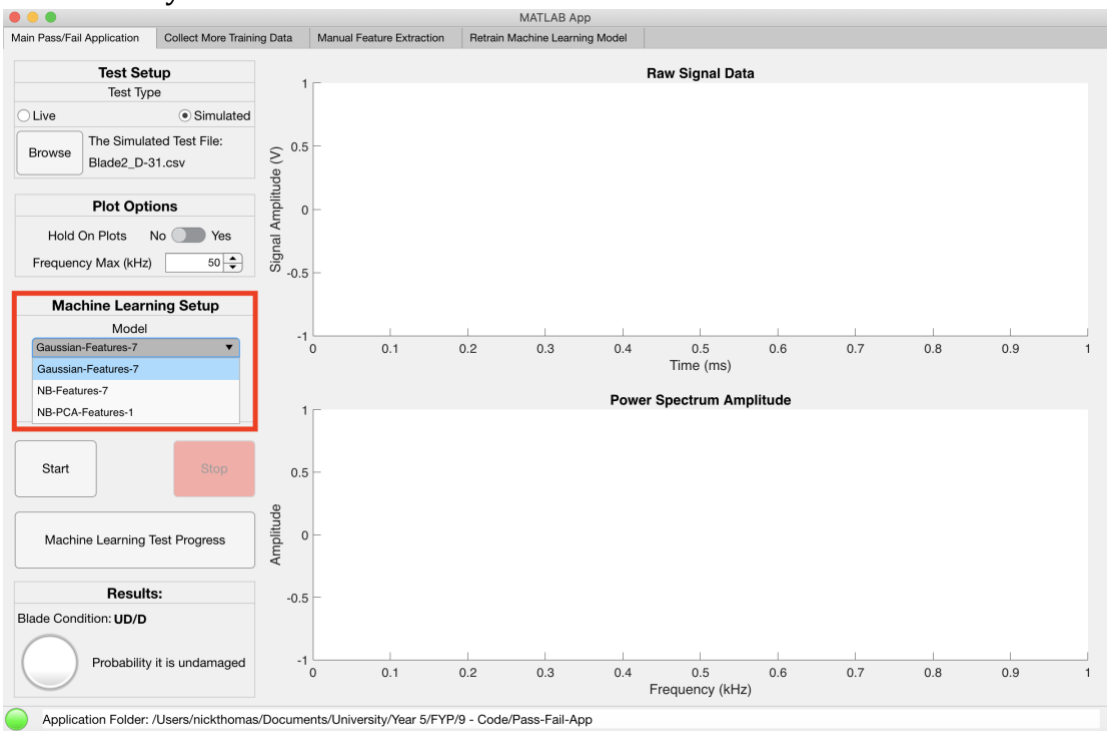
1. If the Figure below is presented on start up, then no machine learning models have been created and no features have been identified; go to Collect More Training Data Tab Guide.



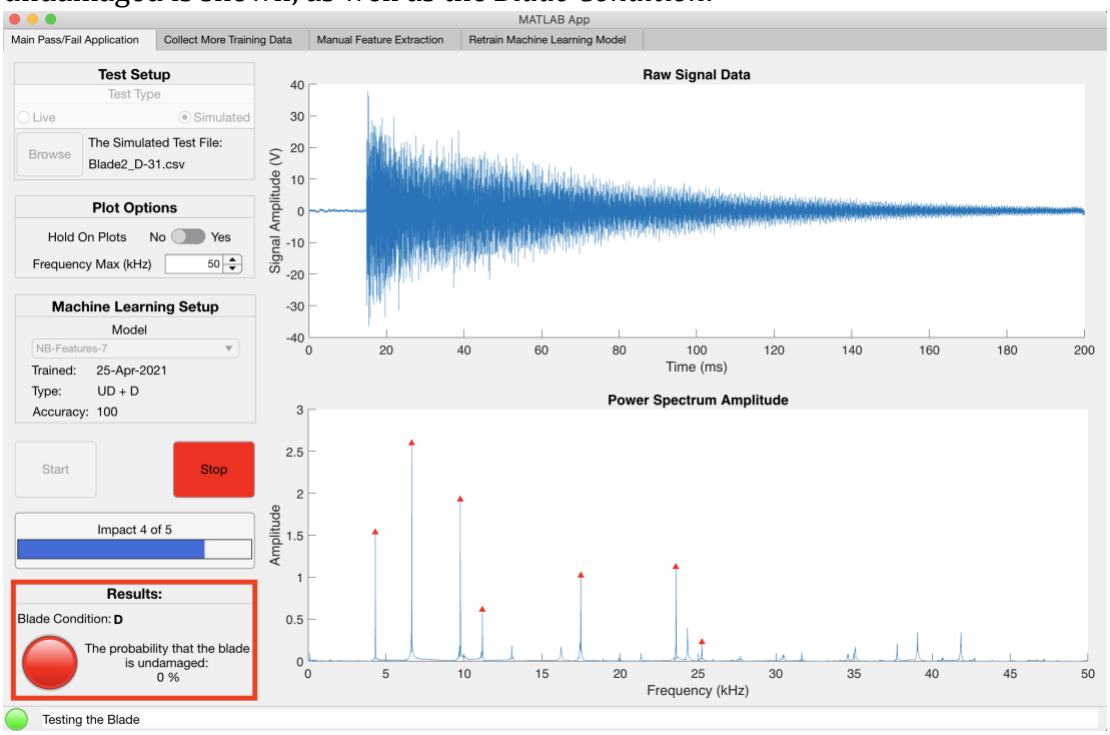
2. If at least one model has been trained, then a classification test can be performed. The *Test Type* radio button can be used to select either a *Live* test using the rig, or a *Simulated* test, as required. If a *Simulated* test is chosen, then the *Browse* button must be used to select an appropriate .csv file.



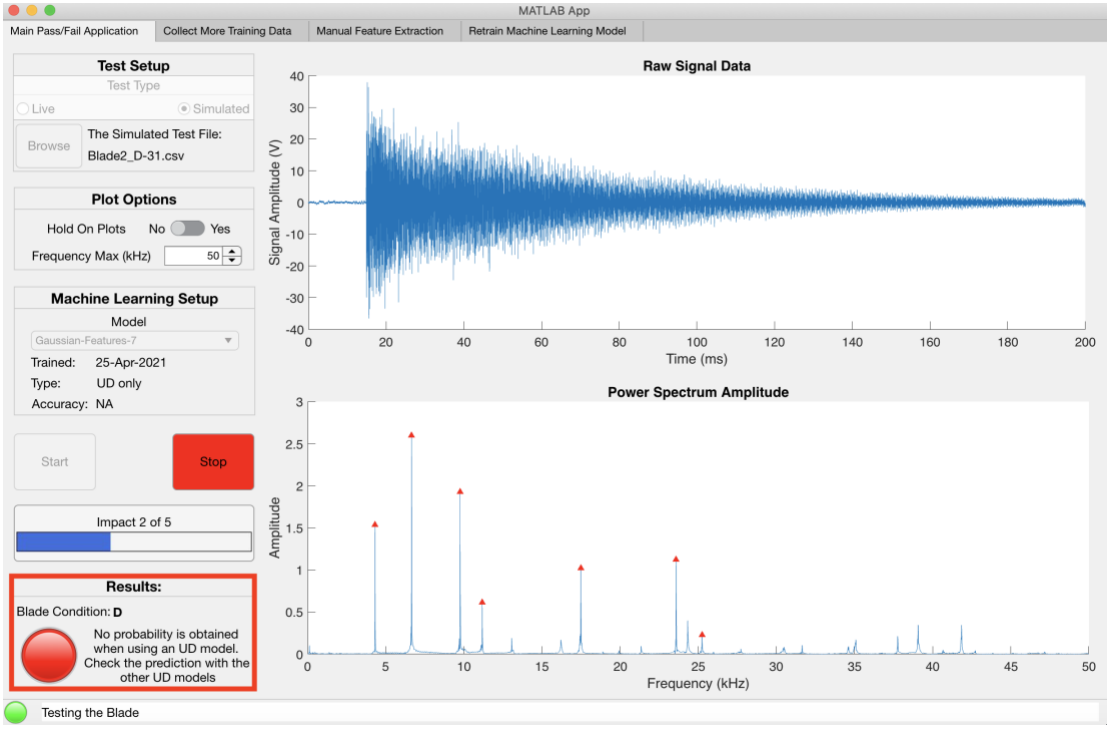
3. The *Hold On Plots* switch can be used if the data from the five separate impacts are to be kept on the plots, or not. In addition, the *Frequency Max* control can be modified to adjust the frequency axis.
4. The *Model* drop down list can be used to select one of the trained models for the classification of the blade. It also displays when the model was trained, its type and accuracy.



5. The *Start* button can be used to begin the classification process. If a supervised model is selected, then in the results section the probability that the blade is undamaged is shown, as well as the *Blade Condition*.



6. If an unsupervised model is selected, then in the results section only the *Blade Condition* is presented.

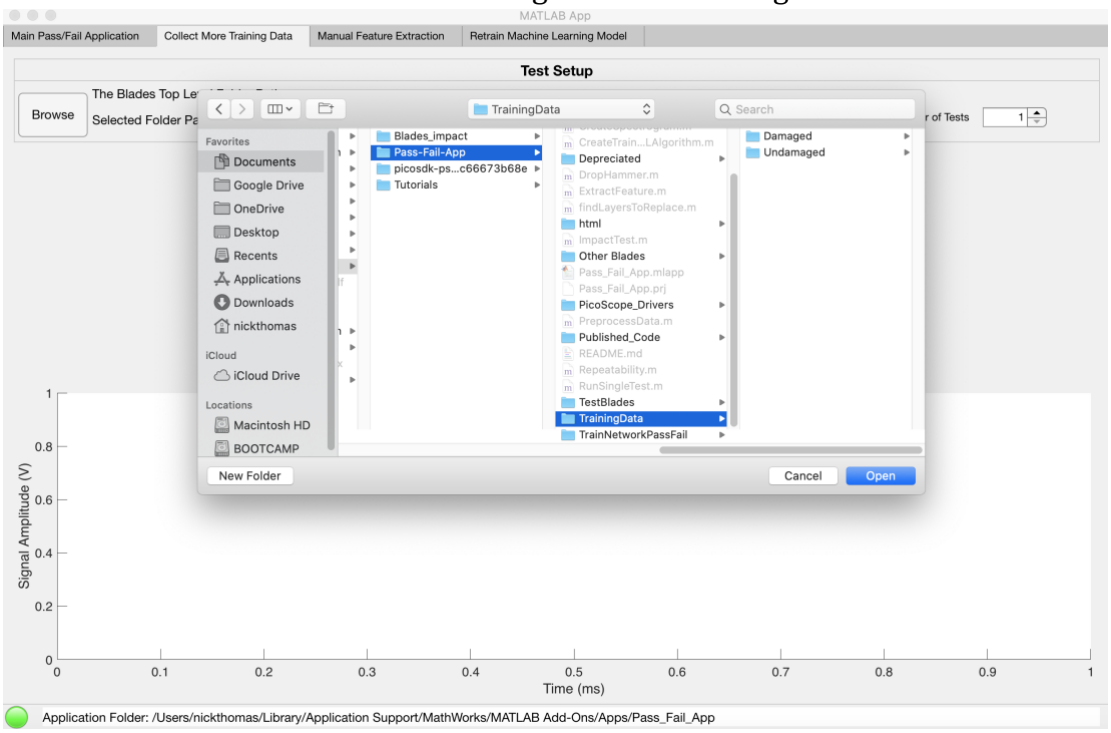


7. For both models, the raw signal and FFT amplitude are presented on the plots. The red triangles represent the values that the machine learning model has identified as being natural frequencies.
8. If a red triangle is not on a peak, then the peak identified in the Manual Feature Extraction tab is not appropriate. This peak should be removed, and the machine learning model must be retrained.

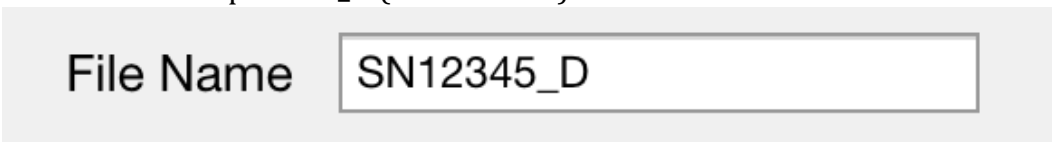
9. Note that this tab can only be utilised if the other three tabs have been used to collect training data, extract the natural frequency peaks, and a machine learning model has been trained.

Collect More Training Data Tab Guide

1. Click on *Browse* and navigate to the folder where the training data is to be saved; this folder must have subfolders 'Damaged' and 'Undamaged' as described earlier.



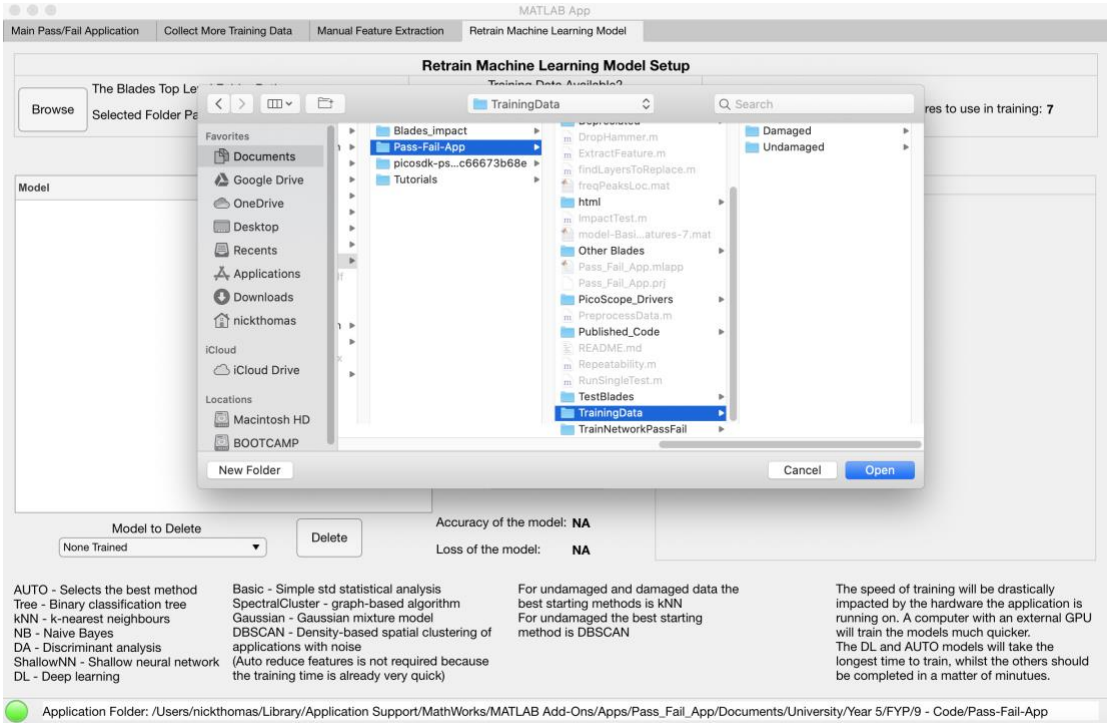
2. Modify the *File Name* so it states the serial number or unique blade identification followed by '_UD' or '_D' depending on the damage state of the blade. These damage conditions must be formatted in this exact way; here are examples of allowable and unallowable names. Please understand the logic of **CheckHistoricalBladeTests.m** if different files names are required. Note, a new folder inside 'damaged' or 'undamaged' will be created with the File Name and will be where all the test CSV files will be kept for that blade name.
    a. SN12345_UD (allowable)
    b. SN12345SmallDamage_D (allowable)
    c. Blade1_D (allowable)
    d. NoTopBlade_D (allowable)
    e. Small5mmHole_UD (allowable)
    f. 5mmHole_UD (unallowable)
    g. 1_D (unallowable)
    h. SN12345-chip-2-mm_D (unallowable)



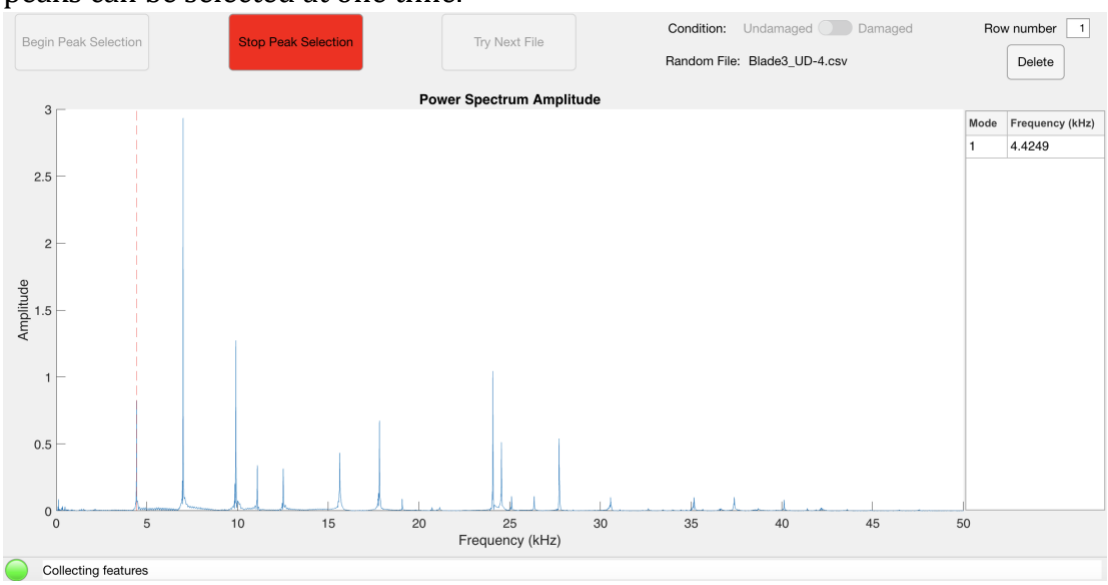
3. Modify the *Number of Tests* as necessary for the number of repeats required
4. Click *Start* to begin the collection. The progress can be monitored by the collection bar. *Stop* can be pressed at any time to stop the collection.

Manual Feature Extraction Tab Guide

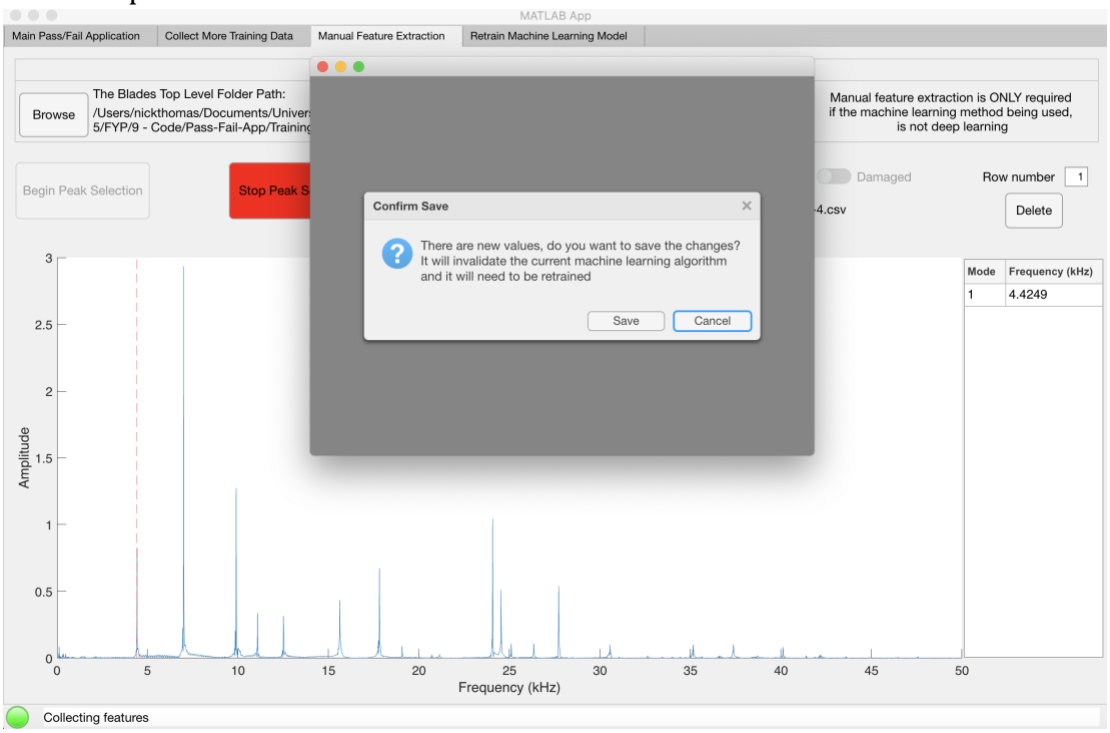
1. Click on *Browse* and navigate to the folder where the training data is saved, this folder must have subfolders ‘Damaged’ and ‘Undamaged’ as described earlier.



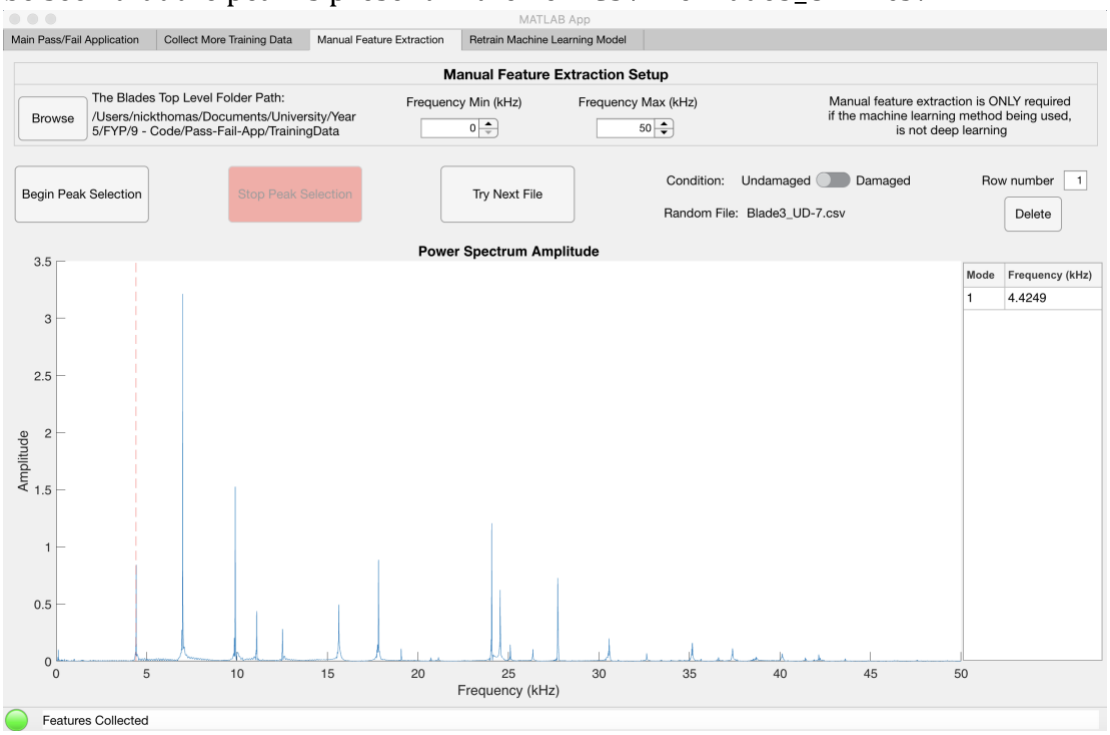
2. Click the *Begin Peak Selection* button to enable the plot to allow the user to interactively ‘click’ the location of a peak. To select a peak, hover the mouse over the tip of a peak which is believed to be the location of a natural frequency. The mouse can be moved after the orange dashed line has been added and the value has been updated in the table on the right. Here, a frequency of 4.42 kHz was selected. Click the *Stop Peak Selection* button to stop selection. Note that multiple peaks can be selected at one time.



3. A pop up will appear if new features have been selected. Click *Save* if you are happy with the selection of new peaks, if not, click *Cancel* which will remove the peaks from the plot and table.

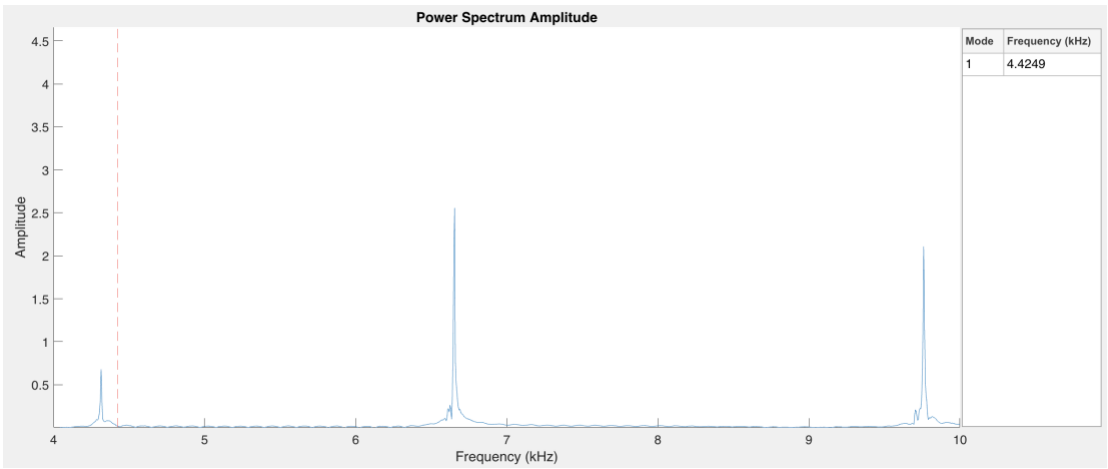


4. This peak is valid for the current repeat which was from the file Blade3_UD-4.csv. The important step is to see if this peak is present in other repeats, identifying that it is indeed an actual natural frequency instead of a reflection. This is a crucial step as it directly impacts the machine learning accuracy. Click *Try Next File* to view the peak on another file. This check should be repeated for all selected peaks. It can be seen that the peak is present in the new CSV file Blade3_UD-7.csv.

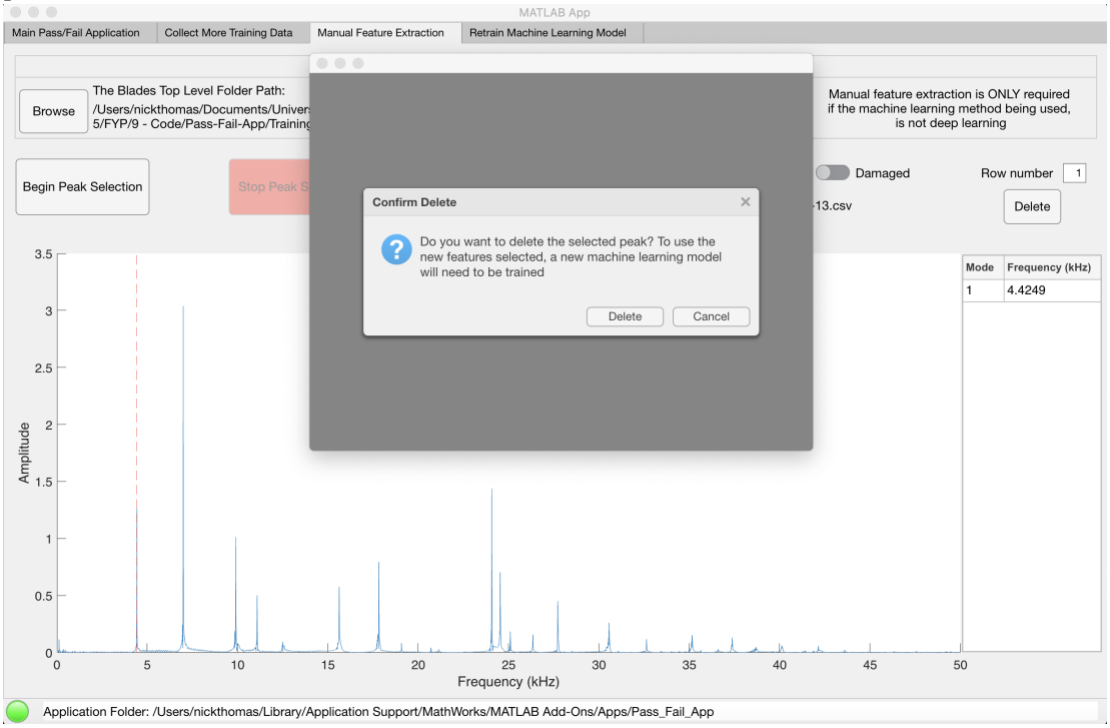


5. Step 4 should be repeated for all the selected peaks and for a reasonable number of different *Undamaged* blade files.
6. The next step is to check that the peaks selected for the undamaged blades match or are in close proximity to the peaks for the damaged blades. Use the *Condition* switch button to change to *Damaged* blades and use the same process as Steps 4 and 5, checking that the peaks align for a reasonable number of files. As expected for damaged blades, the natural frequency is lower by a small amount. The location of the peaks of the other files should be at a maximum of ± 500 Hz away from the orange dashed line and therefore the value in the table. Eg 3.92 – 4.92 kHz.

7. The *Frequency Min* and *Frequency Max* values can be modified so that it is possible to zoom in on particular peaks on the plot to ensure that they are close enough. The plot can still be used interactively, even if the *Begin Peak Selection* button has not been pressed to identify the exact value of peaks, by hovering the mouse over it.

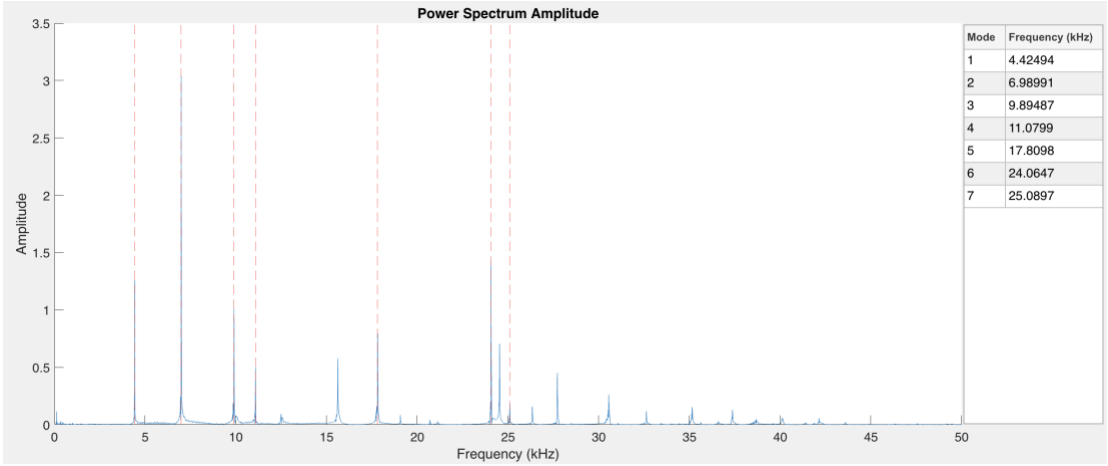


8. Any frequency value can be deleted by modifying the *Row number* value and pressing the *Delete* button above the table. A pop up appears to ask if the user wants the peak value to be deleted. Click *Delete* if happy to remove the selected peak value, if not, click *Cancel* which will cancel the delete.



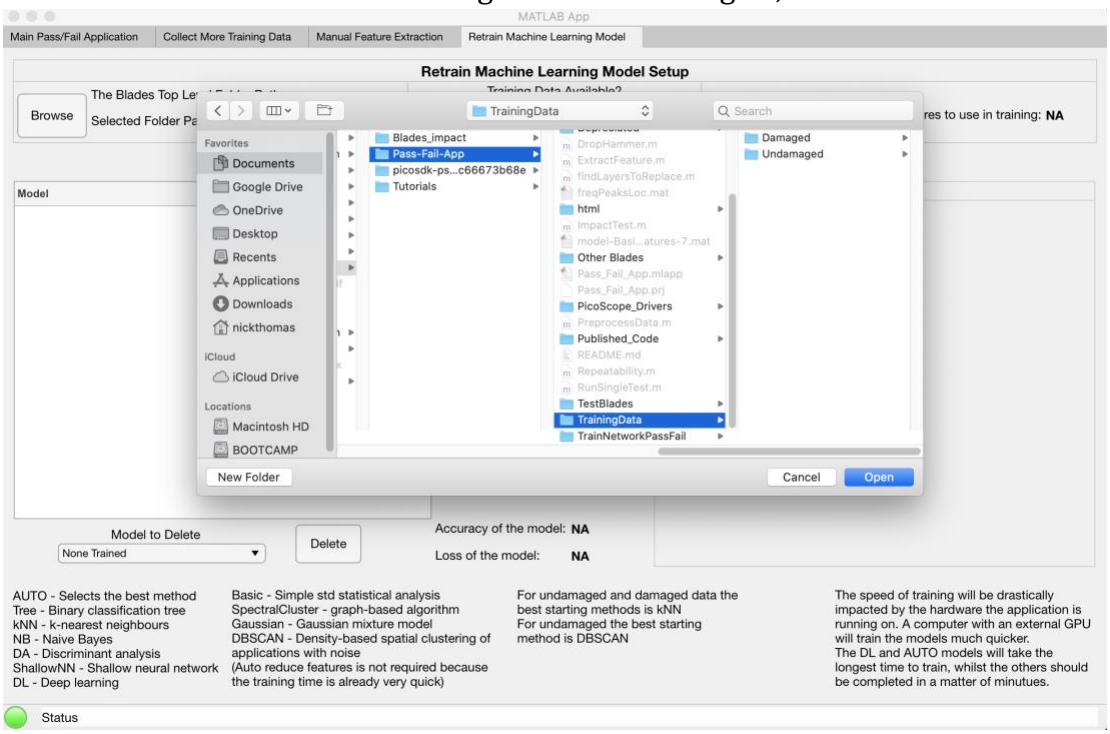
9. Steps 2 – 6 should be repeated until enough peaks have been selected. There is no minimum value of peaks required, as they directly relate to the accuracy of the machine learning model. The accuracy can be identified by the Retrain Machine Learning Model tab. It is suggested that 5 is a good number of features to start with as it is better to add more later, as required.

10. **Note – a good spread of peaks for the tested blades is shown below**
    a. The peaks selected MUST be at least **1000** Hz away from each other
    b. When checking to see if a selected peak is appropriate; for each repeat checked, there MUST be a peak within **± 500** Hz of the orange dashed line

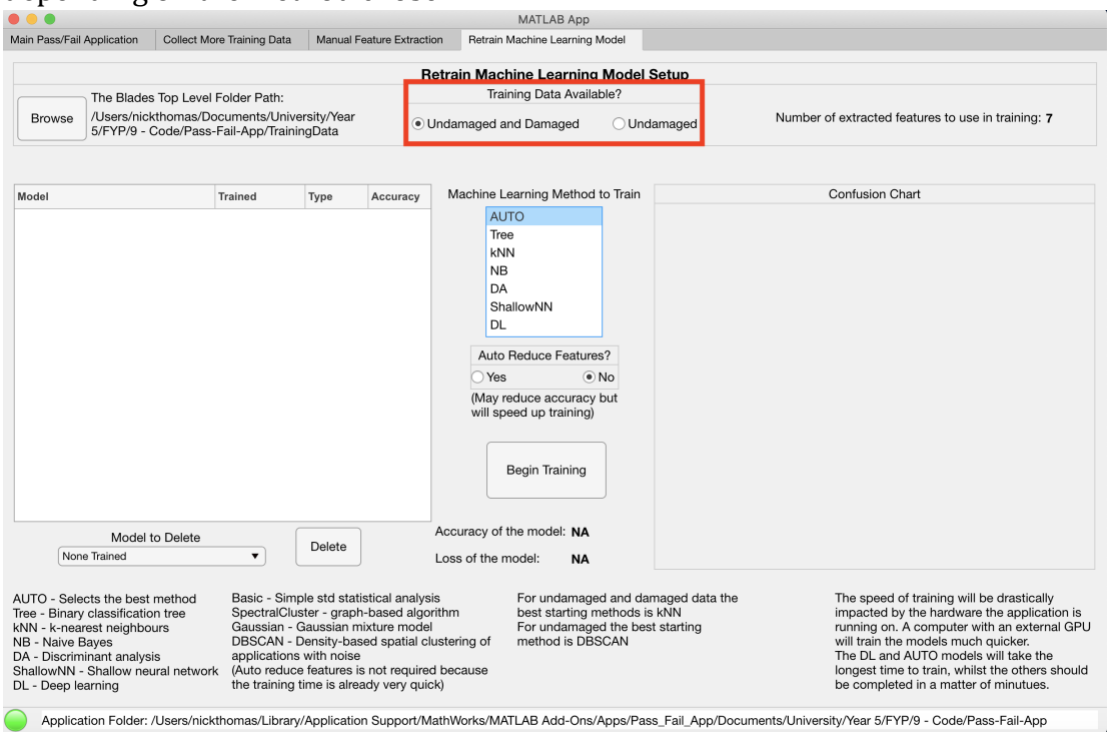


Retrain Machine Learning Model Tab Guide

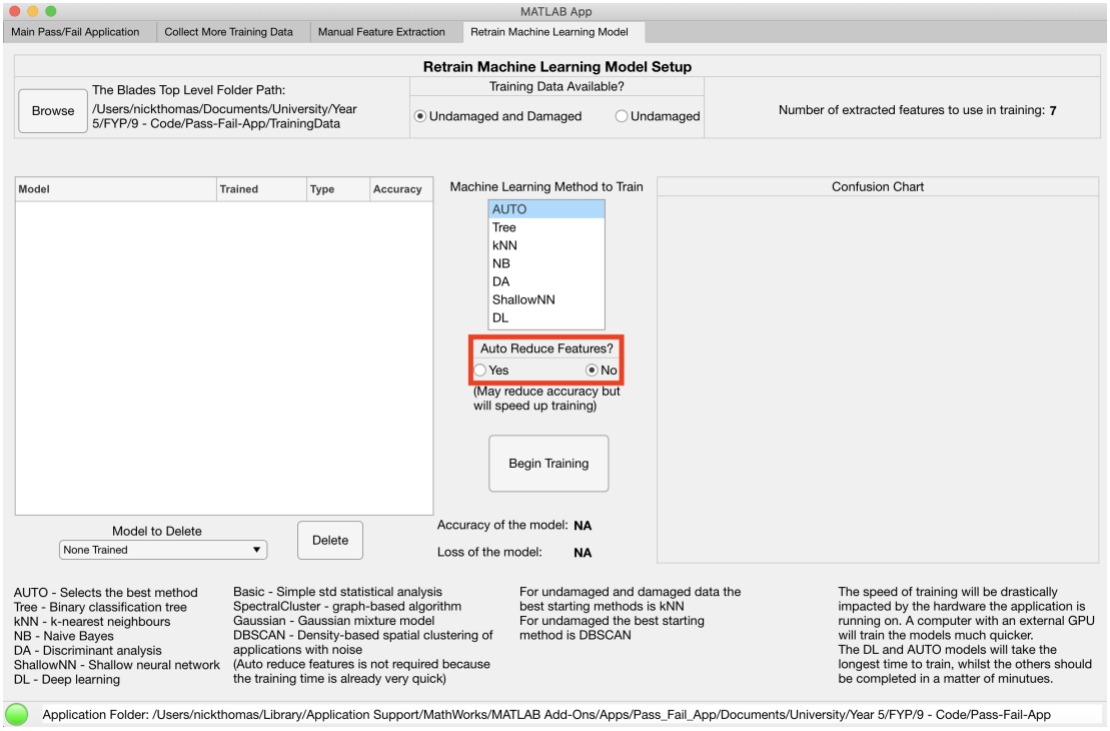
1. Click on *Browse* and navigate to the folder where the training data is saved; this folder must have subfolders 'Damaged' and 'Undamaged', as described earlier.



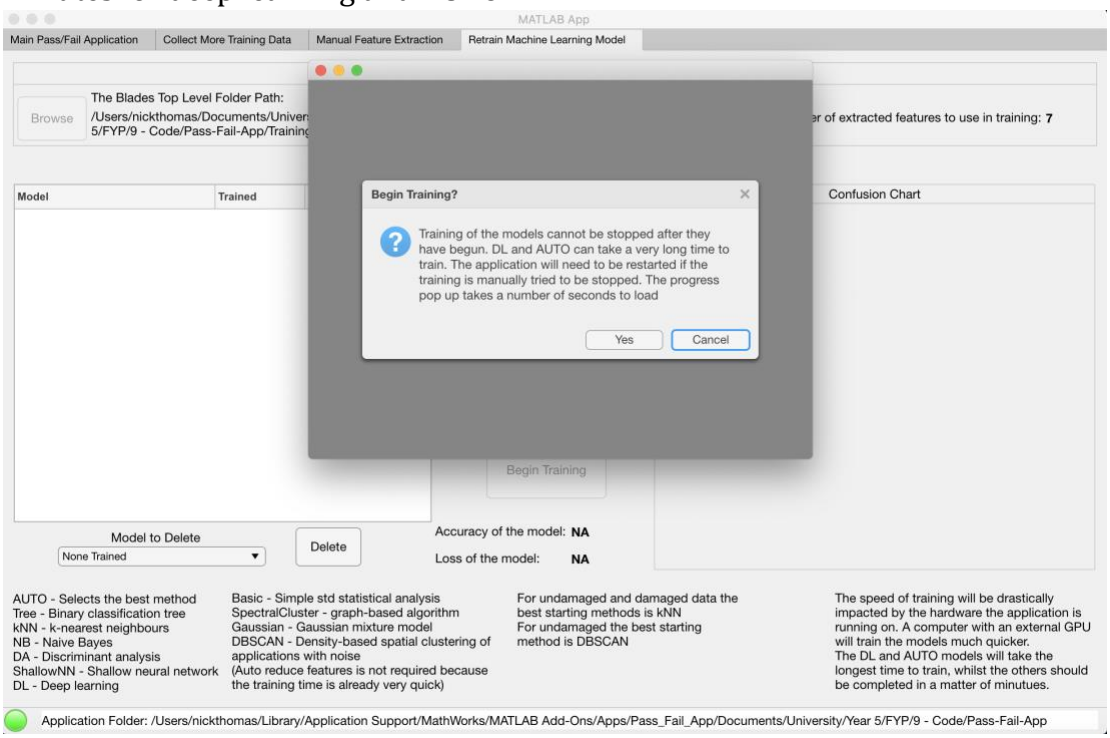
2. Use the *Training Data Available?* radio button at the top to select if you want to train an unsupervised machine learning model which needs *Undamaged* data only, or a supervised machine learning model which needs *Undamaged and Damaged* data. The *Machine Learning Method to Train* list is automatically updated, depending on the method chosen.



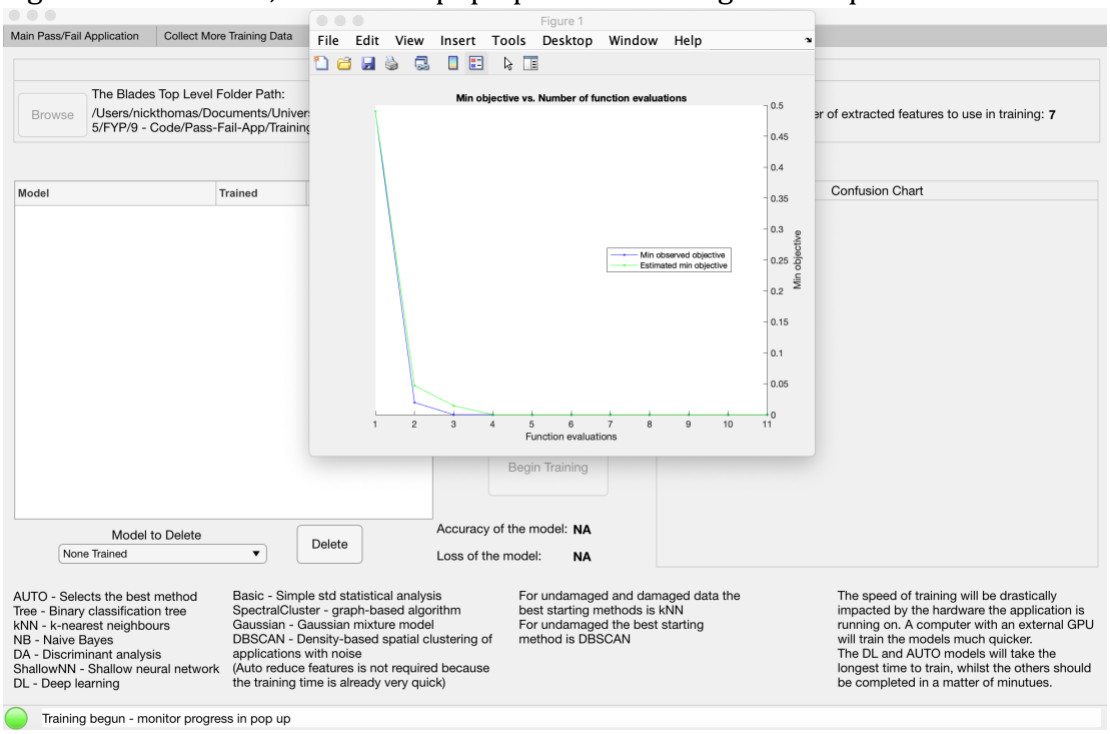
3. If a supervised machine learning model is to be trained, then the *Auto Reduce Feature radio* button can be selected. This performs principal component analysis to reduce the number of features required. It is used to reduce the training time if it is significant but can decrease the accuracy by a small amount.



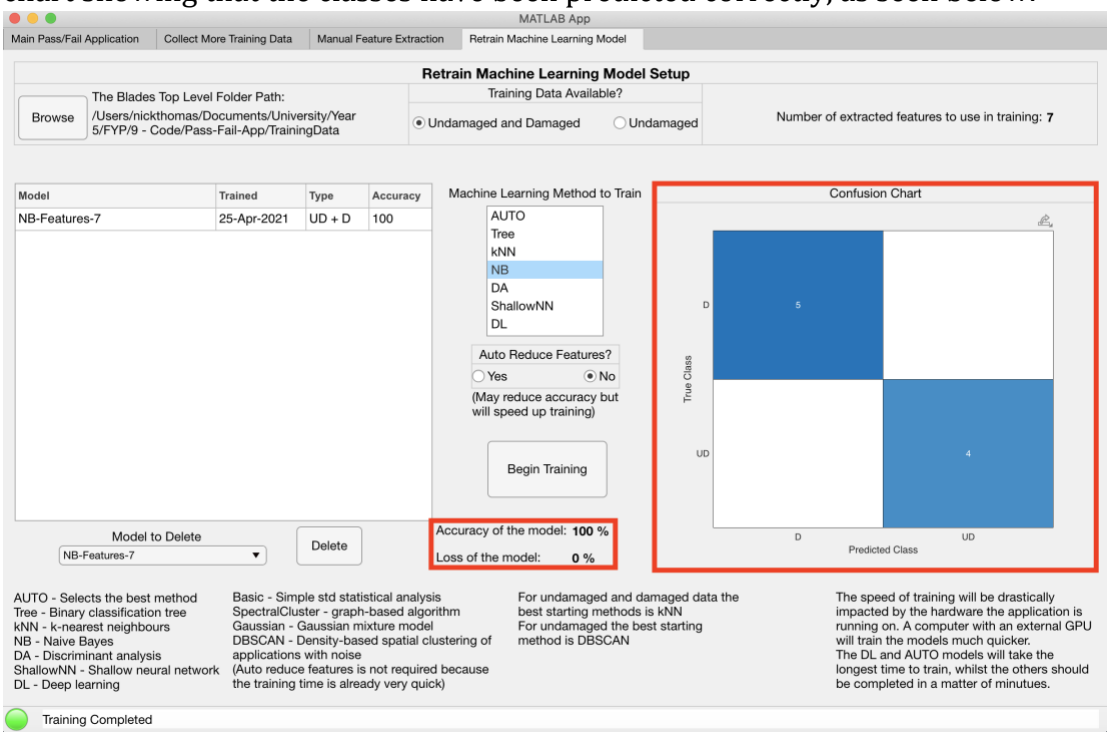
4. The *Begin Training* button can be pressed to begin the training. A pop up box is generated to confirm the model wants to be trained. With a dataset of 60 samples, the training takes approximately one minute for normal algorithms and 10 minutes for deep learning and AUTO.



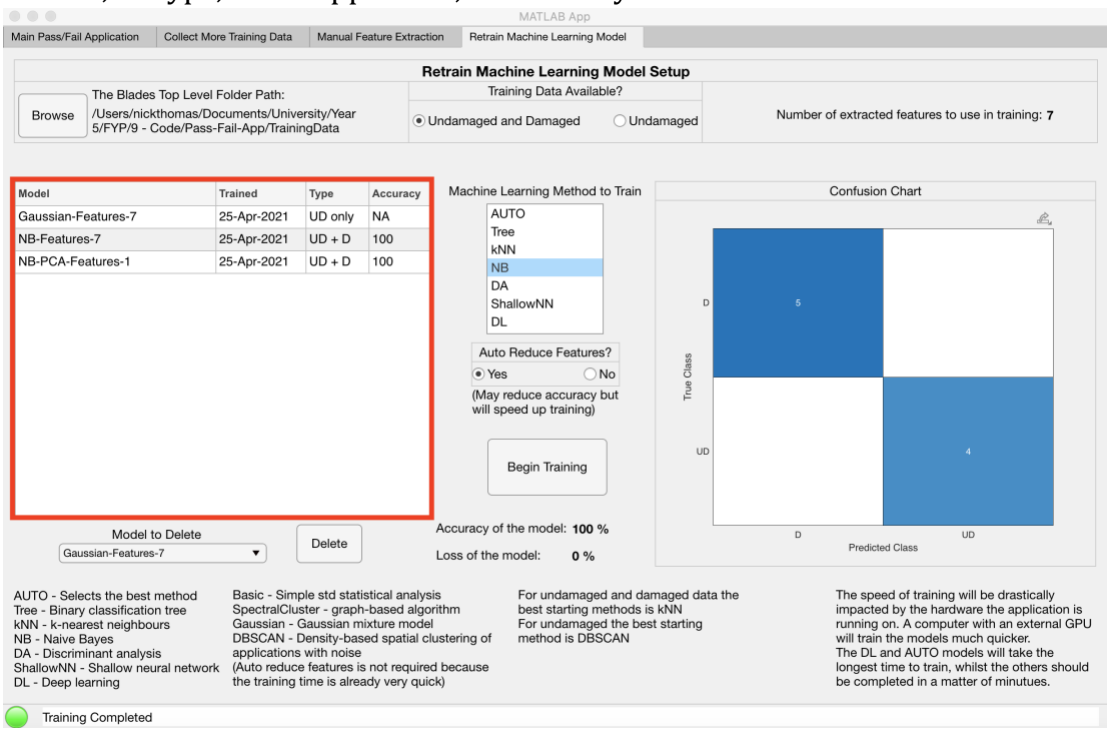
5. A second pop up box is generated if a supervised algorithm is being trained, which can be used to track the training of the model. This usually arrives after 10 seconds, therefore, please be patient. These pop ups are different for each algorithm; an example of naïve Bayes is shown below. If an unsupervised algorithm is chosen, there is no pop up as the training time is quick.



6. If a supervised algorithm has been trained, the accuracy and loss of the model is below the *Begin Training* button, as well as the *Confusion Chart* on the right. The best models have the highest accuracy, the lowest loss values and a confusion chart showing that the classes have been predicted correctly, as seen below.



7. If an unsupervised algorithm has been trained, there are no accuracy and loss values, and the confusion chart is unpopulated.
8. The list of previously trained models can be found in the table on the left. This shows the model's name with the number of features used to train it, when it was trained, its type, and if applicable, its accuracy.



9. The *Model to Delete* drop down list can be used with the *Delete* button to delete previously trained models. A pop up appears to ask if the user wants the model to be deleted. Note that if a new model is trained with the same settings as a previously trained model, it will be overwritten.