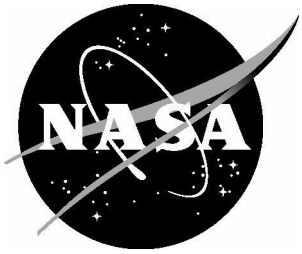
NASA/CR–20240005578



# Presound: UAV Diagnostic System Enabled by Vibration-Based Machine Learning

*Kennan Arlen*
*GreenSight, Boston, Massachusetts*

*Gwenda Law*
*GreenSight, Boston, Massachusetts*

*Andrew DeLollis*
*GreenSight, Boston, Massachusetts*

*J. Gregory McDaniel*
*Boston University, Boston, Massachusetts*

*Sheryl Grace*
*Boston University, Boston, Massachusetts*

May 2024

# NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

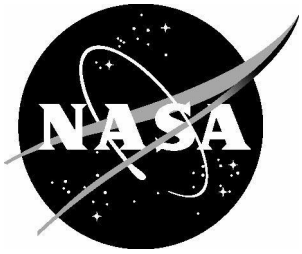
For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- Help desk contact information:

https://www.sti.nasa.gov/sti-contact-form/
and select the "General" help request type.

NASA/CR–20240005578



# Presound: UAV Diagnostic System Enabled by Vibration-Based Machine Learning

*Kennan Arlen*
*GreenSight, Boston, Massachusetts*

*Gwenda Law*
*GreenSight, Boston, Massachusetts*

*Andrew DeLollis*
*GreenSight, Boston, Massachusetts*

*J. Gregory McDaniel*
*Boston University, Boston, Massachusetts*

*Sheryl Grace*
*Boston University, Boston, Massachusetts*

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract 80NSSC21C0559

May 2024

Available from:

NASA STI Program / Mail Stop 050
NASA Langley Research Center
Hampton, VA 23681-2199

**Abstract**

*A low-weight, inexpensive small unmanned aerial system (sUAS) that takes off, performs a mission, lands, and safely stows and recharges itself has myriad future applications ranging from agricultural imaging to last-mile package delivery. Likewise, Urban Air Mobility (UAM) systems will enable people to take air taxis from point to point in cities, rapidly moving commuters long distances without concern for road traffic and congestion. Fully electric aviation systems will be cleaner and quieter than ground transport. Cities could eliminate cars and buses, and convert roads to higher capacity bike and pedestrian throughways. Yet, for sUAS as well as UAM, system reliability and assurance is a limiting factor to deploying affordable autonomous flight systems. For this bright future of aviation to be realized, aircraft must be able to autonomously and accurately self-diagnose health issues both before takeoff and during flight. The GreenSight PreSound system is designed to identify defects on aircraft through intelligent analysis of vibration. It accomplishes this by measuring structural vibrations induced by the vehicle's own propellers, and analyzing that data using a machine learning model that determines whether a defect is present.*

*The PreSound system is designed to require no human oversight, and to operate across a wide array of vehicles through re-training of the model for each target aircraft. PreSound has been developed and seen limited early success using data collected from the GreenSight Dreamer sUAS, a 5lb quadrotor vehicle designed for aerial imaging applications. The final detection model, trained on data with props spinning at 50% throttle, achieves excellent performance with over 99% average accuracy in detecting blade damage using a single FFT vector input. It demonstrates the ability to generalize to new types of blade damage, correctly classifying a different type of blade damage with 98% accuracy. Full test pulses were classified with 100% accuracy, and in live testing, all sets of data during blade movement were classified accurately with over 95% confidence. When trained on in-flight data, the same model achieves an average accuracy of 85% in distinguishing between undamaged and blade-damaged states in flight. The authors believe that these accuracies show significant potential of this approach to expand unmanned flight safety, with significant potential benefits in accelerating Advanced Aerial Mobility (AAM) and UAM aviation applications.*

## 1. Definition of Terms

| | |
|---|---|
| RPM | Rotations Per Minute |
| GPS | Global Positioning System |
| BPF | Blade Passage Frequency |
| FFT | Fast Fourier Transform |
| UAS | Unmanned Aerial System |
| sUAS | Small Unmanned Aerial System |
| IMU | Inertial Measurement Unit |
| PWM | Pulse-Width Modulation |
| HUMS | Health and Usage Monitoring System |
| CNN | Convolutional Neural Network |
| UAM | Urban Air Mobility |

## 2. Table of Contents


**1. Definition of Terms....1**
**2. Table of Contents....2**
**3. Introduction....3**
3.1. System Overview....4
**4. Data Collection Methods....6**
4.1. PreSound-S....6
4.1.1. Presound-S Test Matrix....9
4.2. PreSound-L....9
4.3. In-flight Data Collection....10
**5. Data Processing....11**
5.1. Measurements and Sampling....11
5.2. Moving Window FFT....12
5.3. Different Metrics And Ways To View The Data....14
5.3.1. Transfer Function Ratios....14
5.3.2. Spectrograms and BPF Normalization....15
5.4. In-flight Data Characteristics....19
**6. Model Training and Testing Methods....20**
6.1. Architecture....20
6.2. Data Split....21
6.3. Augmentations....21
**7. Results....23**
7.1. Dot product damage comparison....23
7.2. In-flight Results....24
7.2.1. Presound-L Vs On-Board Comparison....25
7.3. FFT Subsection Testing....26
7.4. Final Detection Results....28
7.5. GradCam Visualizations....29
**8. Discussion....31**
8.1. Model Generalizability....31
8.1.1. Data Splitting System....32
8.1.2. Model Structure....32
8.1.3. Physical Setup....32
8.2. Sample Rate During Streaming....33
8.3. GradCam Analysis....33
**9. Summary....33**
**10. References....34**
**11. Appendices....36**
11.1. Variable Sampling Rate Study....36
11.1.1. Interpolation artifacts....43

11.2. Method for 2D Scaling..... 44
11.3. Sample rate differences with streamed data.....45

# 3. Introduction

System reliability is a limiting factor for deploying truly autonomous, unattended flight systems for both military and civilian operators. Like manned aircraft, to achieve high reliability, these systems require frequent human inspections and maintenance. GreenSight's PreSound system is designed to automatically identify defects during a preflight inspection or during in-flight operation, eliminating the need for human preflight inspections and enabling reliable unattended operation. PreSound utilizes a combination of classical mathematical and innovative machine learning techniques to solve this industry stumbling block.

Current commercial vehicle health monitoring systems, such as Honeywell's Health and Usage Monitoring Systems (HUMS), tend to operate by building a physics based model of how a vehicle gearbox or other specific component naturally vibrates during operation [1]. This model is created through careful analysis of gear meshing, bearings, and other components including characteristic changes that could be expected when the component deteriorates or becomes damaged. Preprogrammed thresholds are then set so the system can give a warning if it detects a vibration outside the threshold in the characteristic areas.

The PreSound system, by contrast, uses machine learning to assess the vehicle holistically. This approach has gained significant traction academically [1] [2] [4], but is comparatively new and not commonly seen in commercial health monitoring systems. By collecting operational vibration data for healthy vehicles as well as vehicles with known damage states, a learning model can be trained to detect the presence of the damage states regardless of the specific physics or components involved - as long as a vibration signature is present for the damage, the model can learn to identify it the same way an object detection network can learn to identify the presence of an object in an image. This approach allows PreSound to potentially be used with any vehicle and any vibration sensor or category of fault, with only a re-training phase necessary for the model to re-calibrate to a different environment. PreSound can even work with the existing accelerometers commonly found in UAV flight controllers

The PreSound detector uses a 1 dimensional convolutional neural network (1D CNN) to detect vibration signatures that correspond to vehicle damage states, a similar approach to that demonstrated in [3] but applied to airframes as opposed to building structures.

Over the course of this Phase II SBIR program, the PreSound system was taken from proof of concept to a prototype that has undergone multiple revisions using real-world data, with several significant obstacles being identified and eventually overcome. Our primary objective was to develop a reliable and accurate preflight check system that can detect various types of prop damage and other vehicle defects, such as loose screws, that could be identified by a human during a pre-flight inspection and may affect the drone's reliability and safety. To achieve this goal, we collected a large dataset of vibration data from accelerometers inside a quadcopter drone operating under different environmental conditions. Various signal processing techniques, including Fast Fourier Transform (FFT), transfer function ratios, and pole/zero analysis, as well as time domain techniques were used to extract relevant features from the vibration data. These features were then used to train a machine learning model to detect damage, using a combination of supervised and unsupervised learning approaches. Each set of extracted features was evaluated to determine how useful they were for detecting damage, and features that did not provide significant detection improvement were discarded.

Average model performance at detecting various prop damage at the time of this report is >98%. This result also sets a precedent for incorporating new categories of vehicle damage into the model which

GreenSight aims to do with its commercial UAV fleet, where GreenSight is planning to deploy the prototype PreSound system in 2023.

The other category of damage tested during this effort was loose airframe screws. A specific screw was loosened from a standard 15 in-lbs of torque to 2 in-lbs, and the model was trained to detect this difference. Though some elements of the analysis showed promise, we were not able to generate a model that reliably detects the loosened screw. Considering how subtle this indication was and the limited amount of data collected we do not find this result discouraging. Additionally, with the success of the blade damage model we expect other variations of screw damage (e.g. a missing screw) may be detectable with a similar approach.

Overall, this program enabled the development of a reliable preflight check system using machine learning techniques. The results set a precedent for incorporating new categories of vehicle damage into the model, which will enhance the system's overall effectiveness and reliability as more data are collected. The following sections of this report provide a detailed account of the system development process, data analysis, and performance evaluation.

### 3.1. System Overview

At its core, PreSound is a system for detecting vehicle faults through vibration analysis, where the vibration is induced via normal operation of the vehicle's motors. During a Phase I and Phase II SBIR program GreenSight developed and tested the system primarily using its own UAS, the Dreamer. However the system is meant to be portable to other vehicles, up to and including those with human lift capacity.


Figure 1: *Dreamer UAS. The GreenSight Dreamer vehicle uses Ardupilot autopilot software [6] and GreenSight Ultrablue flight controller with architecture similar to the Pixhawk [7].*

PreSound works by sampling accelerometer data gathered from one of two sources:

1. On board IMUs from the existing flight controller of the vehicle
2. External IMUs rigidly mounted to the vehicle airframe

The first source is intended to be used with smaller UAS, and was the primary source of data gathered to date. Since it uses sensors and computers that are already present in the vehicle. it doesn't add any extra weight- highly advantageous for a small aircraft. The second method is intended to be used with large airframes and was also demonstrated to provide similar detection resolution to on-board sensors, though it has only been tested on small UAS so far. This was done using a prototype mountable sensor box (named PreSound-L) which also contains an embedded computer to perform the vibration analysis.

| PreSound-S (Small UAS) | PreSound-L (Larger Airframes) |
|---|---|
|  |  |
| Example: GreenSight Dreamer Quadcopter | Example: NASA eVTOL AAM Vehicle |
| • Optimized for small, SWAP-constrained platforms<br>• PreSound-S runs on the UAS' existing flight controller and companion computer<br>• Uses onboard IMU for multi-axis vibration data collection<br>• Adds no weight and hardware cost to vehicle<br>• Requires software integration for use on different UAS platforms<br>• Capable of preflight diagnostic routine using reversed propellers, as well as in-flight monitoring | • PreSound-L is optimized for larger airframes with more complex structures and mechanical subsystems<br>• Vibration data collection and algorithms run on small, inexpensive, dedicated hardware nodes<br>• Each node is essentially a self-contained PreSound system, but data sharing between nodes will allow for additional capabilities<br>• Only requires minimal integration with aircraft's flight control systems, which can be costly for larger, passenger-carrying platforms<br>• Monitoring nodes will be based around GreenSight's compact, low-cost UltraBlue avionics and computing package. |

Figure 2: *PreSound-S vs PreSound-L*

A portable test cradle that the Dreamer vehicle could perform its preflight check in (and take off from) was adapted to be used as a data gathering platform, pictured below. All on-ground data gathered during this effort with a Dreamer vehicle was obtained with some configuration of this cradle:

Figure 3: *Instrumented Dreamer in two versions of the preflight testing cradle (clamped and unclamped) used during data gathering,*

# 4. Data Collection Methods

## 4.1. PreSound-S

All data gathering for PreSound-S with a Dreamer vehicle used some configuration of the test cradle shown in Figure 4. Early in the program several variations were tested, including plastic vs. composite vs. wooden material, and clamped vs unclamped drone mounts. After comparing results from these setups the wooden/composite cradle with no clamping was selected, with no notable difference noted between wood/composite. In the last quarter of the program foam damping at the points where the vehicle arms make contact with the cradle was also added to help generate more consistent data, shown below:

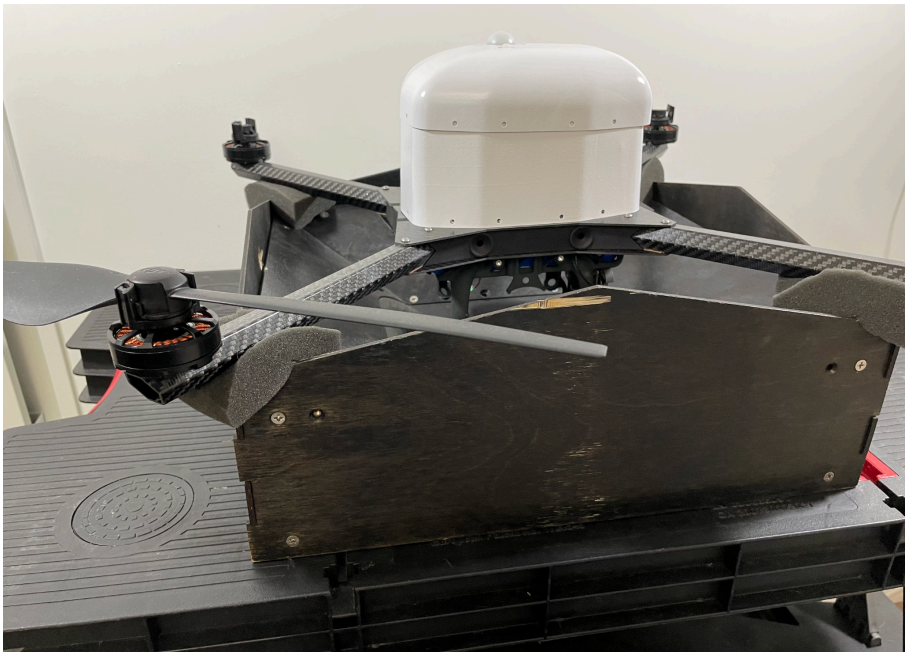


Figure 4: *Test cradle with simple foam isolation.*

During the preflight test process, software running locally on the Dreamer's compute module first places the autopilot system into a motor test mode. A series of test pulses are then sent to the motors, either pulsing each motor in sequence or pulsing all motors together. For this effort 30s pulses were used as the standard, resulting in 30s (for all motors together) or 120s (for each motor in sequence) of motor runtime to gather one "pulse" worth of data for the vehicle. (Note that the motor pulses we use do not generate enough thrust to make the UAV lift off or flip over).

While the motors are spinning, the flight controller IMUs continuously log readings, either directly to the SD card on the flight controller or streamed via a 1.5Mbaud serial link to the Nvidia Jetson Nano compute module [5]. During a motor pulse test the flight controller is configured to log IMU values at the highest supported sample rate, which is 1000-1600Hz for the Ultrablue flight controller depending on configuration.

Once a pulse or series of pulses have been captured to a log file, the data can be immediately used by a trained PreSound model to predict whether a fault is present. When a pulse test is performed with a known damage state induced on the vehicle (e.g. by attaching a damaged prop) the resulting log files can be labeled and used to build a dataset for training or refining a PreSound model for future use on that vehicle type.

Data collection procedures evolved over the course of the program. Initial datasets were all gathered for 5 different health states:

1. No damage
    a. Motor or motors are engaged on a healthy vehicle with no faults present.
2. Major blade fault
    a. Motor or motors are engaged with one prop having at least ¼" cut off the tip:

Figure 5: *Major blade  damage,*

3. Minor blade fault
    a. Motor or motors are engaged with one prop having minor nicks at multiple points along the blade:

Figure 6: *Minor blade  damage.*

4. Major screw fault:
    a. The outer screw securing the arm for the motor currently being pulsed is loosened from 15 in-lbs to 2in-lbs using a torque wrench

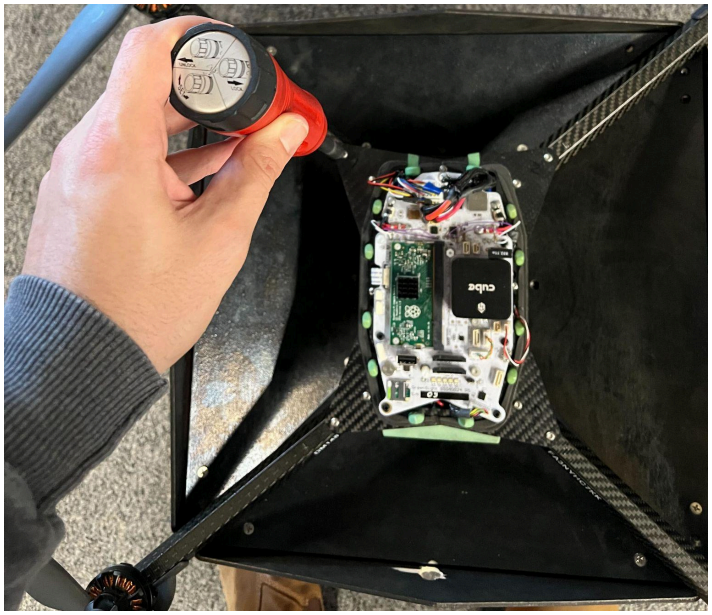


Figure 7: *Uncovered Dreamer UAS outer arm screw being adjusted with a torque wrench.*

5. Minor screw fault
    a. Same as major screw fault except the screw is loosened on the clockwise arm from the one with the active motor (this state is skipped when all motors are pulsed together).

#### 4.1.1. Presound-S Test Matrix

For each configuration, this table gives the number of pulses collected to form one full ‘dataset’ for the purpose of model training:

| Motor | PWM | No Damage | major blade | minor blade | major screw | minor screw |
|---|---|---|---|---|---|---|
| 1 | 1500 | 20 | 5 | 5 | 5 | 5 |
| 2 | 1500 | 20 | 5 | 5 | 5 | 5 |
| 3 | 1500 | 20 | 5 | 5 | 5 | 5 |
| 4 | 1500 | 20 | 5 | 5 | 5 | 5 |
| All | 1500 | 20 | 5x4 | 5x4 | 5x4 | n/a |

Table 1*: Preflight Test Matrix. Note that major/minor distinctions for screw damage are not used for the all motor test. For this test the loose screw is cycled through all arms and labeled as major for each.*

This procedure was used to collect 2 full datasets (several partial datasets were also collected). The PWM value is a digital input value used by the flight controller to control motor throttle level, and is calibrated to range from 1000 (motor off) to 2000 (max throttle). Normally this is mapped to a controller input, but for these pulse tests a precise PWM control value was sent to the flight controller using a software test script running on the vehicle. The PWM control value corresponds to the speed at which the motor spins during each pulse, with 1500 PWM corresponding to ~50% of maximum throttle, generating ~3000 RPM for the Dreamer UAS test vehicle. Later data gathering efforts shifted to only using 5 pulses of no damage (from 20), as the experimental results did not differ significantly when restricting the training to only use 5 pulses.

### 4.2. PreSound-L

For larger aircraft such as a helicopter or e-VTOL, using accelerometers from a single internal flight controller is not sufficient to gather vibration data across the entire airframe: multiple distributed monitoring nodes are beneficial in this case. At the same time, adding some small hardware components is a less onerous proposition on a larger aircraft as opposed to a tiny UAS. To address the large-aircraft use case GreenSight developed a prototype ‘PreSound-L’ hardware node which consists of the same Ultrablue flight controller board and Nvidia Jetson Nano compute module used internally on the Dreamer aircraft, that can be rigidly mounted to the interior or exterior of  a vehicle. The PreSound-L node contains 3 embedded IMUs, and also supports external IMUs which can be mounted separately from the main node and attached via SPI. During this program GreenSight tested the Presound-L system by attaching it externally to a Dreamer aircraft:

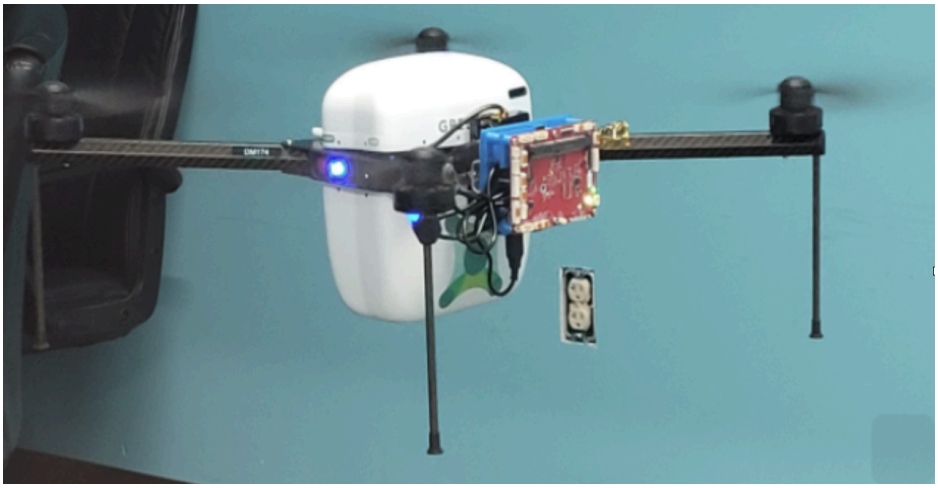


Figure 8: *Dreamer flying with attached prototype PreSound-L node.*

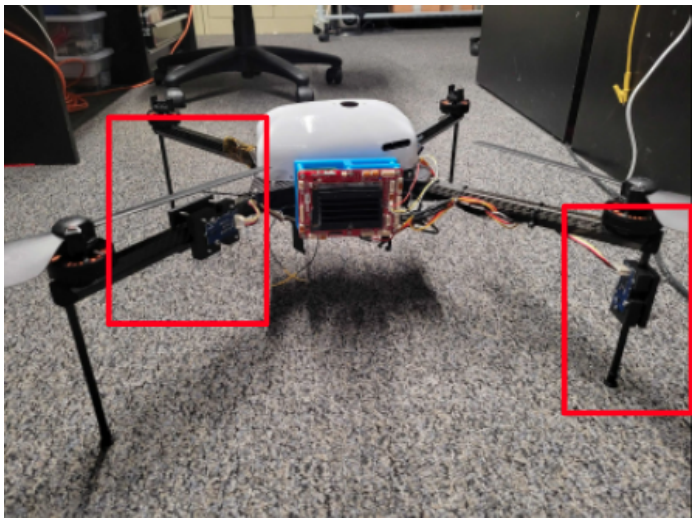


Figure 9: *PreSound-L node with additional attached IMUs for vibration measurement at specific points.*

### 4.3. In-flight Data Collection

A procedure for gathering data from a dreamer UAS in flight was also tested. The purpose of the in-flight data gathering was to measure and analyze the differences in vibration signatures between a flying vehicle and one being tested on the ground, and to perform a preliminary evaluation of how well the PreSound model would be able to detect defects during flight as opposed to on the ground under controlled conditions. Continuous in-flight monitoring can potentially provide advance warning of an impending failure and allow the aircraft to take corrective actions such as landing.

Multirotor vehicles in particular pose several challenges that need to be overcome for analyzing in-flight vibration data, when compared to data captured pre-flight, notably:

1. A consistent motor RPM is difficult to target, since the vehicle must continuously adjust motor speeds to maintain flight.

2. Even at times when motor RPM is not changing significantly, the RPM can still vary widely between motors depending on vehicle weight distribution and other variables.
3. It is not possible to pulse motors individually in-flight - the only 'mode' obtainable is all motors, as compared to pre-flight where each motor can be pulsed independently

Despite these challenges we were able to devise a procedure that allowed gathering usable data for detecting damage in-flight, though the noise level is higher than data gathered on the ground. By repeatedly hovering the vehicle in an indoor no-wind environment, enough controlled variables are established to make the data usable without requiring many hours of data collection, which would otherwise be necessary to cover the full range of potential in-flight conditions.

The procedure for in-flight data gathering used the following steps:

1. Vehicle was flown in an indoor test chamber
2. Using a manual controller vehicle was brought to a hover ~6-8ft above the ground
3. Hover was maintained for 60s
4. Vehicle landed and props were allowed to spin down for ~15s

These steps were repeated 5x for each damage state, resulting in a total of 300s of hover vibration data for each damage state per full dataset. In all 3 full and 3 partial in-flight datasets were collected.

# 5. Data Processing

The underlying rationale for the initial development of Pre-Sound stemmed from the notion that signals obtained from vehicle measurements should exhibit changes in frequency content corresponding to variations in the vehicle's state, such as the emergence of defects. In this context, both on-board vibration and contact pressure sensing, as well as nearby acoustic sensing, were taken into consideration. We began our investigations by comparing the effectiveness of acoustic sensors, contact microphones, and accelerometers at detecting defects. The results of these early experiments indicated that accelerometer vibration data alone was sufficient for detecting damage; overall accuracy of the detection models improved only negligibly (<1% in most cases) when combining all modalities, as compared to using the vibration data in isolation. This was a fortunate result, since accelerometers are present in every UAS to support flight control systems, whereas the other sensors would need to be added.

The accelerometers within the UAV's IMU were employed to capture temporal vibration signals along three axes, denoted as Accel X, Accel Y, and Accel Z. In conventional vibration analysis, a hammer-type excitation is typically applied to a structure, whereas PreSound uses the rotation of one or multiple rotors to excite the structure. In the context of a pre-flight check, when the vehicle is on the ground rotors can either be turned slowly, or in reverse, so that they don't generate enough thrust to flip over the UAV. It is also possible to run each rotor individually in a sequential execution of the motor pulsing protocol as outlined in Table 1.

## 5.1. Measurements and Sampling

We developed a data collection procedure that generates log files containing a series of 30 second pulses of vibration data (See section 3.1.1). The three acceleration signals for such a series of pulses is shown in Figure 10. In order to work with this time signal and generate a related frequency spectrum, some processing steps are necessary.

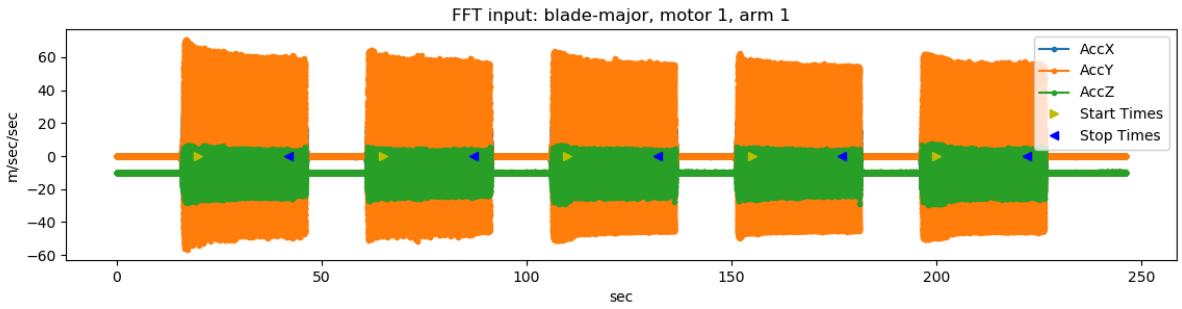



Figure 10: *5 Raw motor pulse data plotted in the time domain. The yellow and blue arrows indicate the sections of each pulse which will be extracted for further analysis. Sample rate: 1000Hz.*

As discussed in the previous section, the source of the acceleration data was the UAV's on board flight controller. Unlike a dedicated high-performance vibration logging device, the flight control computer does not record vibration samples at a consistent time interval since it is doing many other tasks besides logging vibration. When a sensor cannot sample consistently at a specified sampling rate, the time signal must be resampled onto a uniform time step or an adaptive type of FFT must be used to obtain an accurate spectrum. figure 11 shows sampling rate variation for an example data file. We investigated the performances of different methods for obtaining the FFT from a signal with nonuniform time sampling. More specifically, we tried (i) interpolation techniques including linear interpolation, Fourier transform resampling, and sinc interpolation for generating uniform samples, as well as (ii) directly taking an FFT from the non uniform samples using a function called `nufft` (non uniform FFT) implemented in MATLAB. The details are shown in Appendix A. The study determined that linear interpolation was the ideal solution.

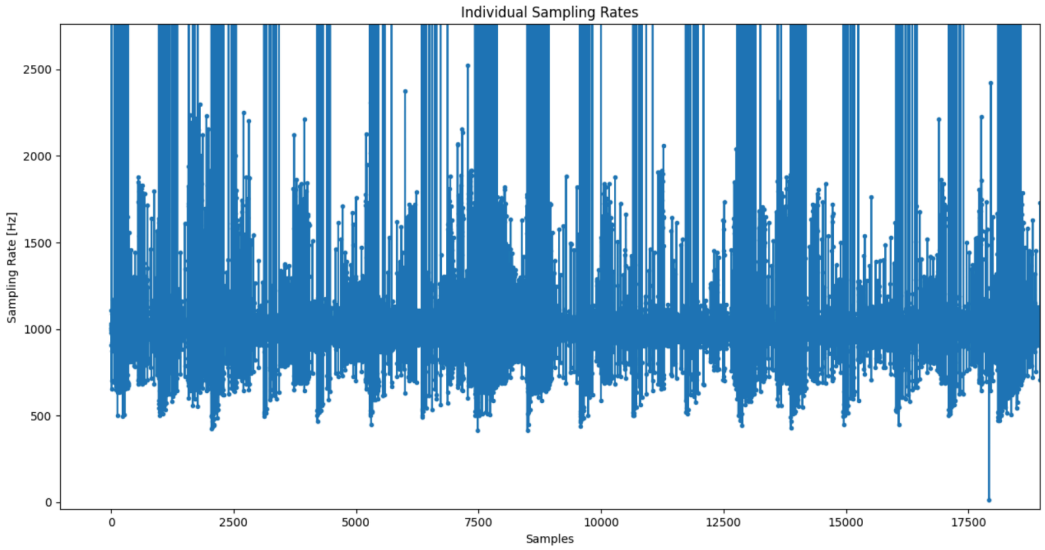



Figure 11: *Sampling rate variation of a sample data file.*

## 5.2. Moving Window FFT

Following the resampling of the data, the associated frequency spectrum must be obtained. A careful examination of this process is outlined in Appendix 9.1, emphasizing the importance of minimizing any potential impact on the final FFT. Once that was assured, efforts were directed towards exploring effective means of utilizing the entire acquired dataset in a meaningful manner. Specifically, the utilization of ensemble averaged FFTs to discern data variations was studied. The resulting spectrum exhibits a

remarkable degree of clarity when the ensemble averaging approach is adopted. To illustrate, by applying a moving window technique wherein the starting time point is shifted by 500 samples, approximately 60 FFTs are generated from a single pulse, each comprising 3000 partially overlapping data points. By performing this procedure for every pulse, a total of 300 FFTs can be obtained per data file. Subsequently, the ensemble average is calculated by averaging all the FFTs for a particular damage state, yielding a representative composite spectrum that can be visually interpreted and compared.

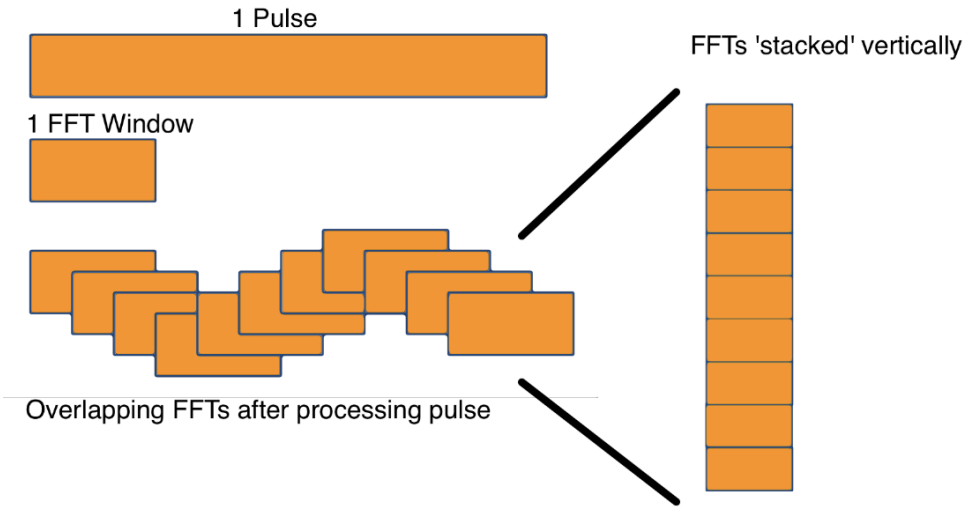



Figure 12: *Illustration of generating a column of FFT data via a moving window.*

A comparison of the ensemble average damage state FFTs is shown in Figure 13. This figure overlays the magnitude of the average FFT for every damage state on the same figure, plotted on a log scale, as well as black lines showing +- 1 standard deviation from the average for the undamaged state. This highlights both the macroscopic spectral changes for different damage states, as well as how large those differences are when compared to the normal expected variation.

While the ensemble averaged values are shown in multiple figures, for input to the machine learning algorithm it was determined that each individual FFT used to obtain the ensemble average would be concatenated as shown on the right of Figure 12.

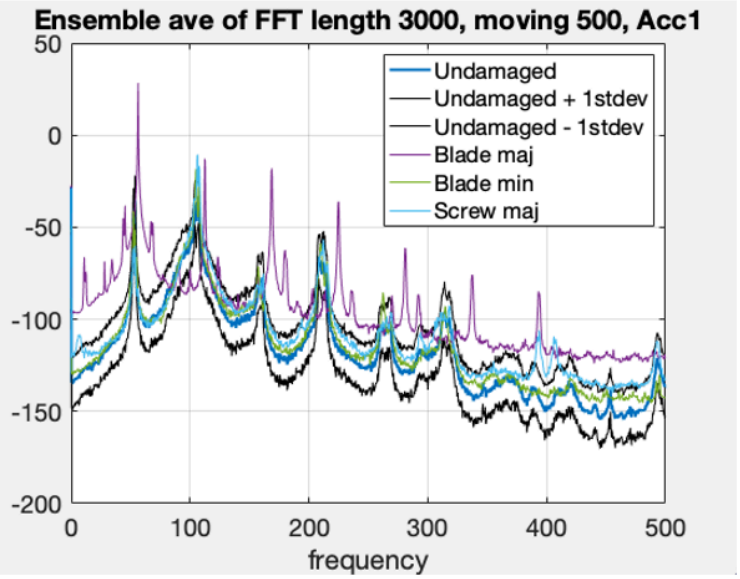



Figure 13: *Comparison of damage state FFTs. The Y axis is the log of the FFT magnitude.*

## 5.3. Different Metrics And Ways To View The Data

Additional metrics beyond the raw accelerations and FFTs that are often used in vibration analysis were considered. Transfer function ratios and spectrograms in particular are discussed in this section.

### 5.3.1. Transfer Function Ratios

Transfer functions are derived using ratios of the magnitudes of the FFTs, e.g. $\frac{fft_{mag}(AccX)}{fft_{mag}(AccY)}$ gives the X/Y transfer function. Transfer functions for X/Y, X/Z and Y/Z were calculated and used as additional vectors for training the ML models, alongside the unaltered FFTs.

Transfer function ratios are used because they help to filter out global bias, and effectively amplify any signal that is asymmetric across the X/Y/Z dimensions. One of the hypotheses made while designing the PreSound system is that many damage types or fault states would have an asymmetric impact on vehicle vibration, and therefore should show up more clearly in transfer function ratios. In comparison, natural changes to vibration due to varying operational conditions or motor speeds are more likely to be symmetric, and would be filtered out. An example of this would be a higher motor power increasing the vibration amplitude equally in X Y and Z - when examining a transfer function ratio, the signal would remain unchanged. By contrast a single damaged or unbalanced prop is likely to disproportionately increase or decrease the vibration along some axes more than others.

An example of the comparison of a transfer function for the different vehicle health states is shown in figure 14. Notably, the variation in TF ratios (indicated by the shaded area) is greater than the difference between damage states. In the machine learning methods, Section 5, it will be noted that these transfer functions were not used when training our final demonstration network.

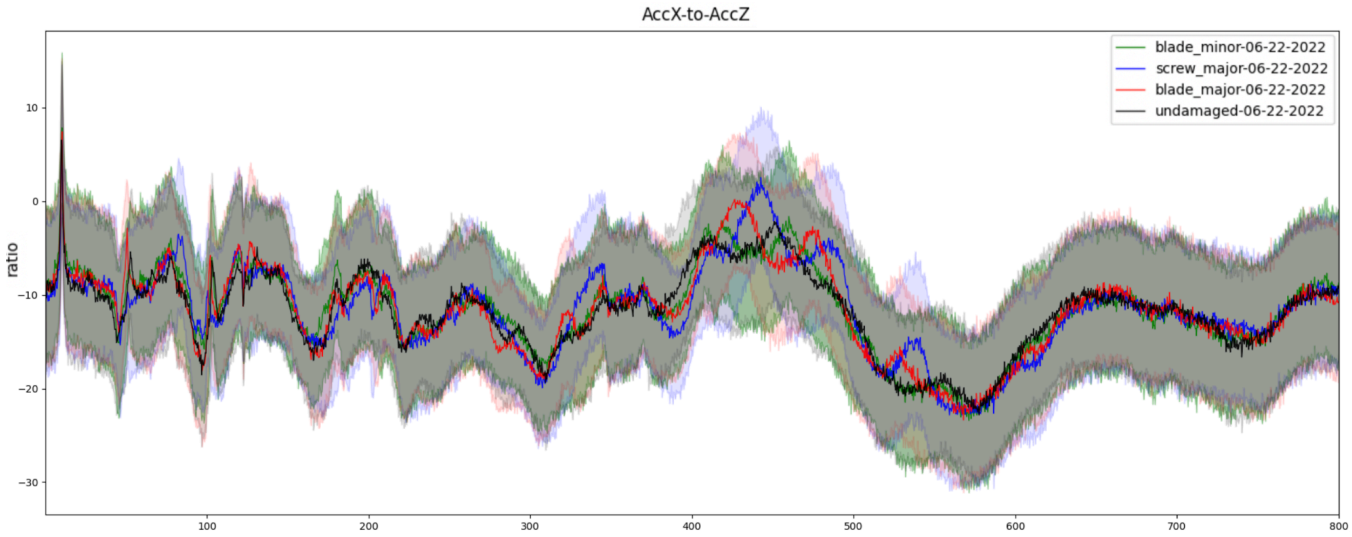



Figure 14: *Example transfer function ratios for ensemble averaged FFTs for each damage state. Shaded region indicates +- one standard deviation for each TF ratio. Horizontal axis is Frequency and runs from 0-800Hz, Vertical axis is the ratio between the average magnitude of the AccX and AccZ FFTs at that frequency.*

### 5.3.2. Spectrograms and BPF Normalization

Another analysis tool that proved useful was the spectrogram. Since our FFTs are generated using a moving window, it is of interest to stack them and consider if there are any obvious changes in the spectrum as time moves forward, especially between motor test pulses. Figure 15 a & b show a composite spectrogram for two cases, prop and screw damaged. To generate these plots multiple pulse tests for each damage state, taken across multiple days, are each converted into a spectrogram. Each individual spectrogram for a given damage state is then stacked vertically to create a plot that helps highlight differences from pulse to pulse within a given test, as well as day to day variations between data collection runs. One notable variation is the shifting of the Blade Passage Frequency (BPF), which is a multiple of the frequency at which the motor props are spinning and changes between runs due to inconsistencies with motor controls and battery charge levels. The variation can most easily be seen at the boundary where spectrograms from different collection runs are attached.

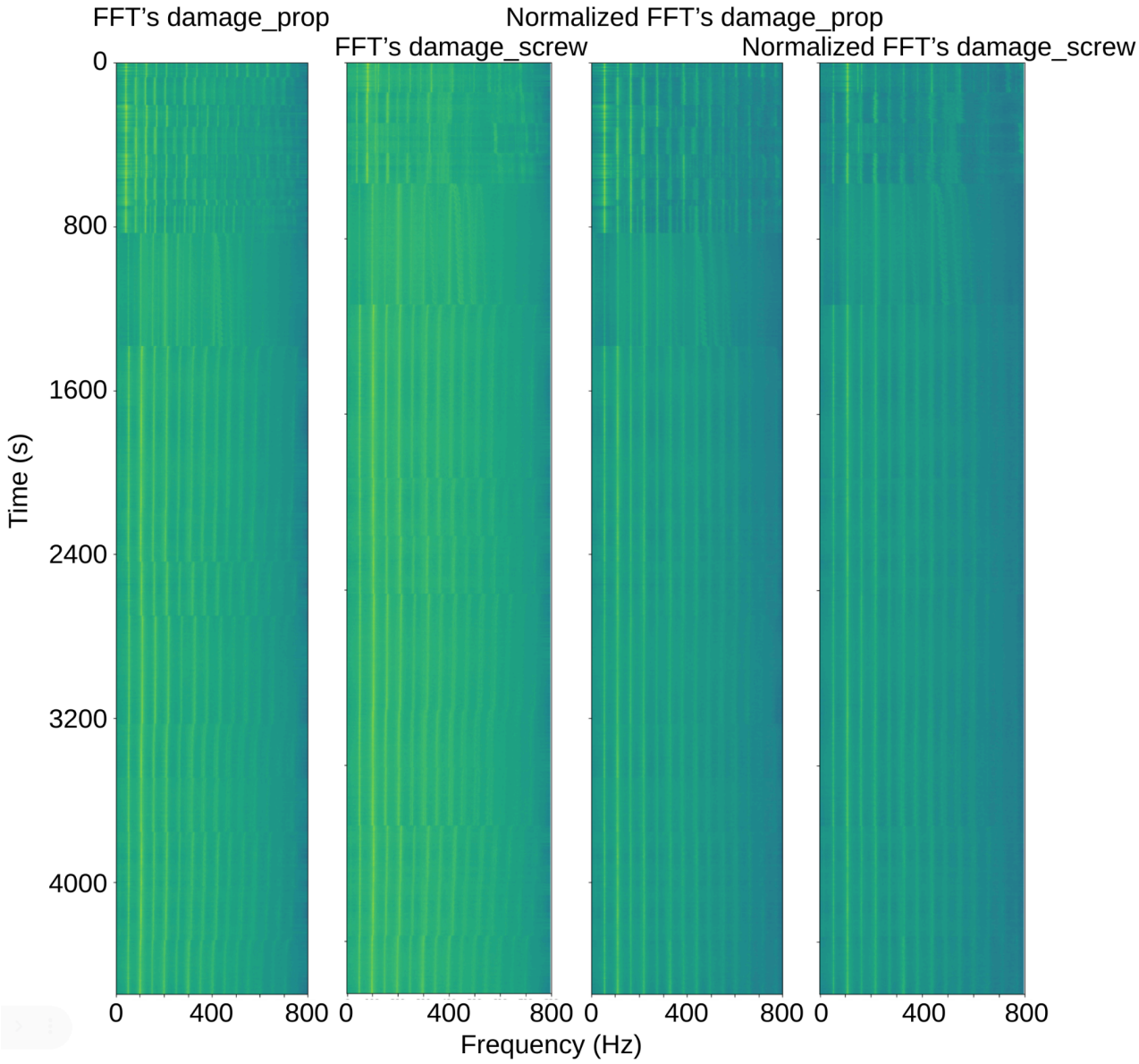



Figure 15: *Spectrogram plots of input FFTsignal during pulses without BPF normalization on the left (a,b), and with BPF normalization on the right (c,d). Note the peak lines on the right deviate much less over time.*

To normalize the input by BPF and remove this variation, a function was developed. Its purpose was to determine the BPF and stretch the FFT. The process involved selecting a target reference frequency slightly above the actual BPF. By smoothing the input and examining the spectrum within a specific range, the second frequency peak, corresponding to the BPF, was detected. An interpolation function was then applied to stretch each set of pulses until its second peak aligned with the target reference frequency. This rescaling procedure ensured that the apparent frequency of the second peak remained constant, achieving alignment across the collected datasets while maintaining the relative spacing between peaks. See more detail in Appendix 9.2. This procedure works well, as seen in c & d in Figure 15. However there are still differences in the data day-to-day based on the amplitudes and noise.

The spectrogram was also used to perform a more detailed comparison of data taken on different days. The plots below provide an example of these variations. The spectrograms from several different tests on

the same day, noted as the 28th, are quite similar. There are large differences though across days, noted as 20th, 22nd, and 28th. Each of these tests were done with the same system configuration with undamaged rotor blades. This day-to-day variation that appears in the data was crucial in informing the machine learning training method, and will be discussed in Section 7.1.

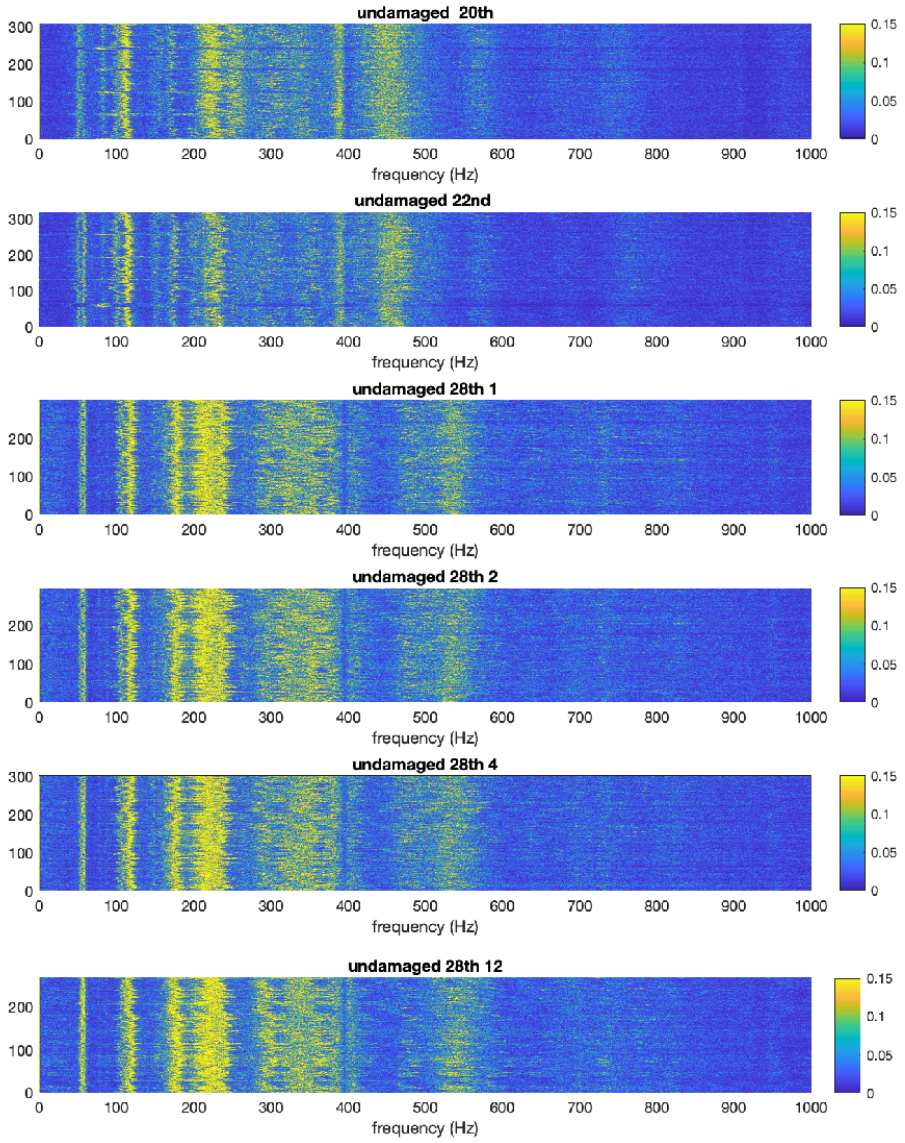



Figure 16: *Spectrograms highlighting the variance in undamaged data collections on different days.*

The spectrogram also helped identify differences between methods of on-ground data collection that were used in Phases 1 and 2 such as clamping or not clamping the vehicle to the test fixture. Clamping introduced extra peaks into the FFT data that would change in frequency over time as seen on the right in Figure 17 (particularly clear between 1000 and 1500Hz). This led to adopting the unclamped configuration as the standard for the test procedure.

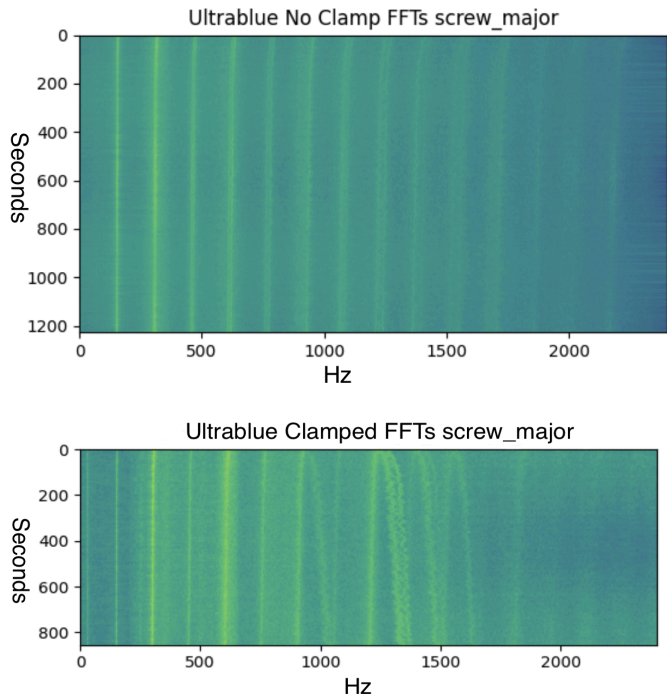



Figure 17: *On ground, all motors spinning with no clamp (left) and clamp (right).*

Finally, spectrograms highlighted how increasing the motor throttle from 25% to 50% during on-ground testing amplified the blade minor damage signal. It is worth mentioning that when blade minor damage was present at 50% throttle, the frequency peaks exhibited significantly less stability compared to the undamaged state or the same damage state at 25% throttle.

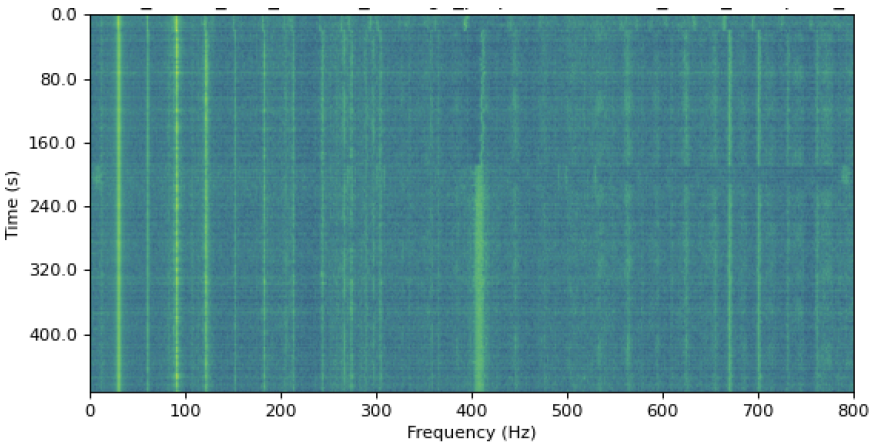



Figure 18:: *Spectrogram of blade minor damage at 25% throttle.*

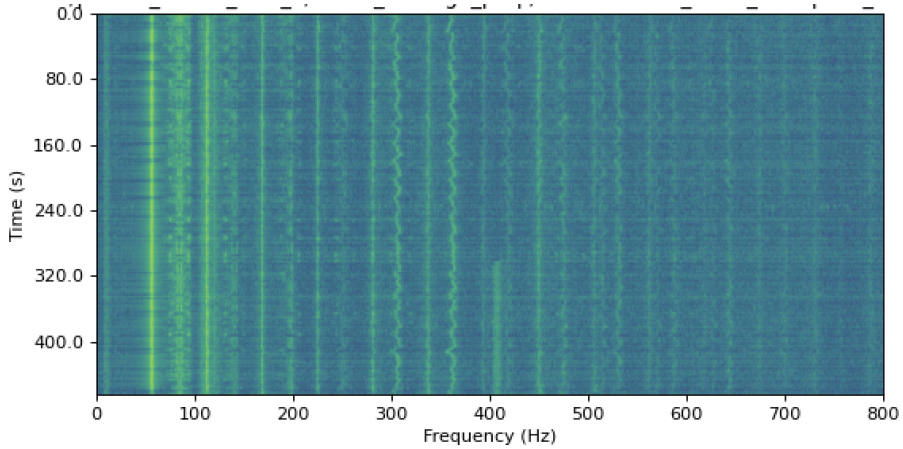



Figure 19: *Spectrogram of blade minor damage at 50% throttle. Distortion ('waviness' of peaks) is much more pronounced.*

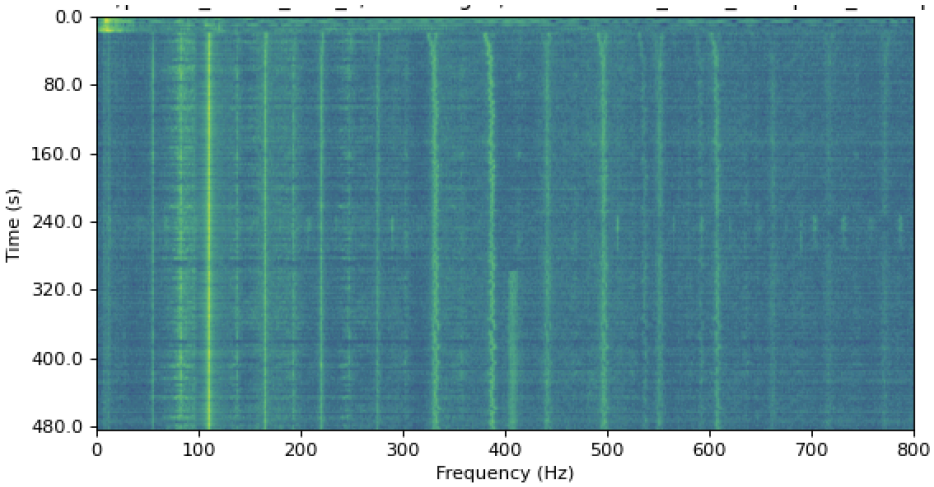



Figure 20: *Spectrogram of no damage at 50% throttle, for reference.*

Another noteworthy observation in these plots is the peak at approximately 400 Hz, which tends to appear primarily in the second half of the spectrogram. These spectrograms were gathered with only one motor spinning at a time, and the extra peak is caused by the three other idle motors having a behavior where they emit regular audible beeps when left armed but idle for over 300s. This behavior is undesirable, however, models trained on datasets with the extra peak present still performed well when tested on data with or without the extra peak so datasets with the 'beep' peak present were not discarded.

### 5.4. In-flight Data Characteristics

The primary focus of our data analysis revolved around the preflight information, during which each rotor could be independently activated and examined. However, when employing fault detection techniques during in-flight operations, it becomes necessary for all rotors to be in motion, thereby eliminating the possibility of guaranteeing specific rotation rates. Consequently, it is unreasonable to anticipate the presence of distinct tonal components at the BPF within the vibration spectrum.

The frequency domain of the in-flight data exhibits higher noise levels compared to the on-ground data.

While this was expected, the level of added noise during in-flight hover states turned out to be larger than anticipated. One contributing factor is the considerable variability in motor RPMs during hover, with for example motor1 spinning up to 40% faster than motor3 during stationary hover. Analyzing the in-flight vibration data becomes more challenging due to the constant shifts in motor RPMs, making it difficult to gather multiple FFTs across a large time window for noise reduction through averaging.

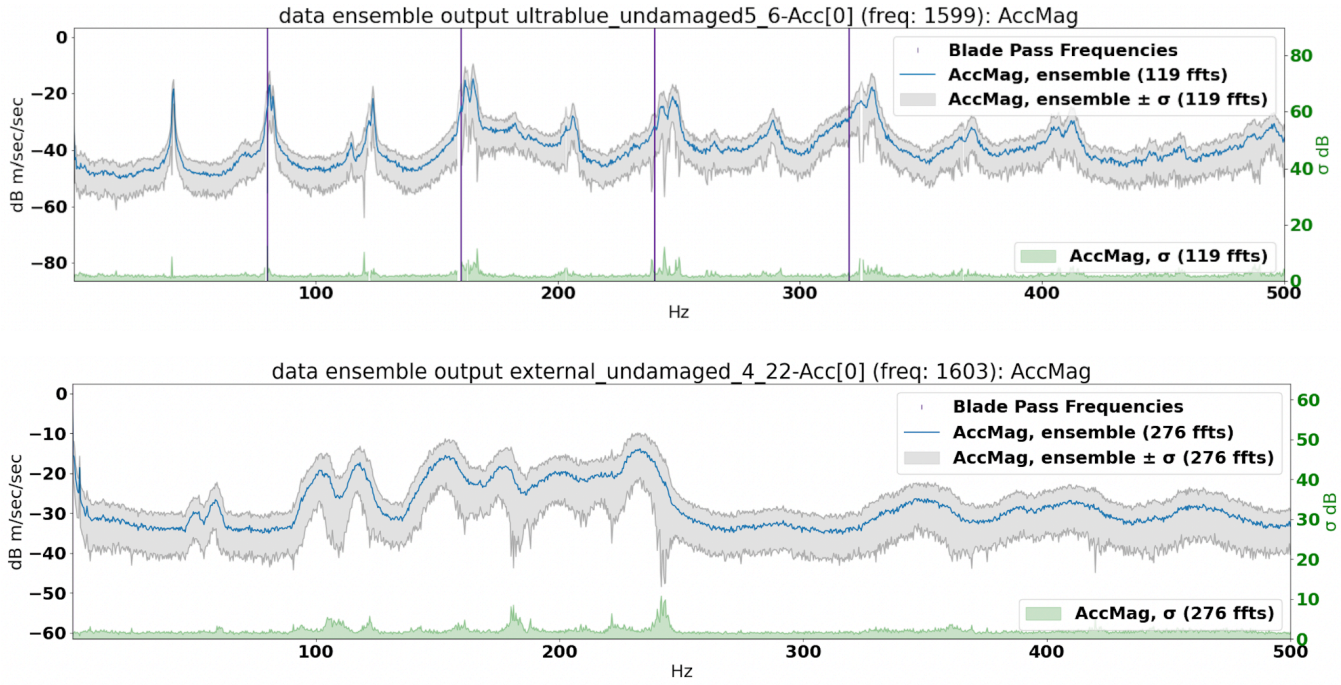



Figure 21: *On-ground ensemble average frequency spectrum (top) vs in-flight ensemble average frequency spectrum (bottom). Purple lines represent the blade pass frequency (~80Hz) and harmonics, where peaks are expected.*

# 6. Model Training and Testing Methods

## 6.1. Architecture

The machine learning model utilized in this project is a 1D convolutional neural network (CNN) that is designed to predict damage states. The network comprises ten layers, which include pooling and dropout layers. Three convolutional layers are employed, with the first layer using a small 3-bin kernel and the others using larger 20-bin kernels. The number of channels increases from 4 at the input layer up to 32. The output from the convolutional layers undergoes max pooling, resulting in only the most important output being used. Finally, three fully connected layers downsample the results to the number of classes. Dropout layers are incorporated throughout the network to prevent overfitting, although they are only used during training and are turned off during inference.

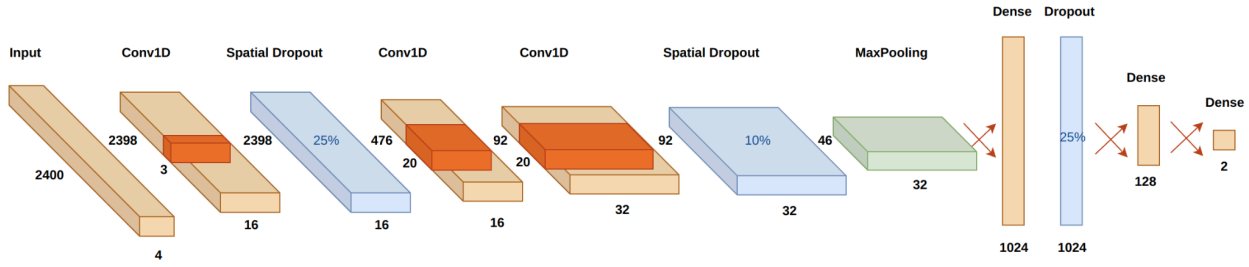



Figure 22: *Architecture of the 1D convolutional network.*

## 6.2. Data Split

To train the model, the data must be split into training, validation, and testing sets. During the first half of this program one method was used for splitting, while in the second half a different method was used.

In the first half, all of the 30s pulses in a dataset were divided by cutting up each individual pulse using a 20-60-20 ratio:

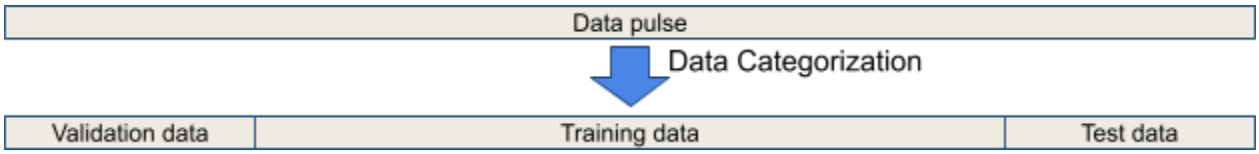



Figure 23: *Splitting the data into 3 categories for validation, training, and test.*

With this method the performance of the network could easily be evaluated against a single dataset (as defined in Table 1). While this method provided useful information and was helpful in refining the models to better interpret the data and achieve high accuracy, it was eventually determined that this approach did not isolate the test data from training sufficiently for the model to be able to generalize well - results achieved on a test set split in this manner would often not be repeatable when tested against an entirely new dataset. This issue and the steps taken to correct it are described in section 6.1: Model Generalizability.

Eventually a second method was settled on and used for the latter half of the program. With this method the pulses are kept whole, and data is instead split into train/test/validation by date, ensuring that each subset contains bin files captured from a dataset not found in any other subset. With this approach a normal training cycle will use validation and test data gathered on a completely different day from the data used to train. This ensures that the generalizability of the network can be verified both during training and afterward. The model is trained with augmentations on the training data during each epoch of training and tested on the validation data. Training is stopped when the performance on the validation data ceases to improve, typically around 10-20 epochs. Finally, the model is tested on entirely separate testing data to ensure its ability to generalize well to data gathered on different days. Although a large amount of data was collected throughout the program, it is spread over fewer days than ideal for testing. As such, cross-validation was performed, involving re-training from scratch using different randomized divisions of the data, to ensure the results remain consistent.

## 6.3. Augmentations

Augmentations refer to the application of various transformations or modifications applied in a randomized manner to the training data to create new, slightly altered versions of the original samples. By generating new samples through augmentations, we effectively increase the size of our training dataset.

The aim is to expose the model to a wide range of diverse and randomized conditions during training, in order to encourage it to learn higher-level features and generalizable representations that are applicable in the target domain.

Certain augmentations can teach the network to be invariant to specific transformations. For example, by applying random rotations to images, a network could learn to identify objects irrespective of their orientation. This helps in reducing overfitting and generalizing better to unseen data. Other popular image transformations include translation, scaling, cropping, flipping, and changing the brightness or contrast of images. All of these transformations don't prevent a human viewer from being able to identify the images themselves.

Applying augmentations to accelerometer data is more difficult, as which transformations mask the part of the signal that indicates damage versus mask unimportant aspects of the signal is unclear. If augmentations are chosen correctly, we can create a neural network that can handle a wider range of variations, generalize better to unseen data, and exhibit improved performance and robustness in classification tasks, especially in scenarios where the target domain differs from the training domain.

Two levels of data augmentation are utilized during a PreSound training cycle. The first is applied directly to the vibration signal before the FFTs are taken and consists of Gaussian noise, high-pass filters, and low-pass filters. These augmentations are applied once, doubling the amount of input data. The second round of augmentations is applied at each batch of training, with the signal being modified differently for each batch. These augmentations consist of stretching the FFT for simulation of different blade RPM frequencies, flipping the signal upside down, adding broad spectrum noise to change the baseline of the FFT, blacking out a random 200 Hz section of the spectrum, and adding a false peak at a random frequency. These augmentation techniques enhance the robustness of the model and improve its generalizability to real-world data.

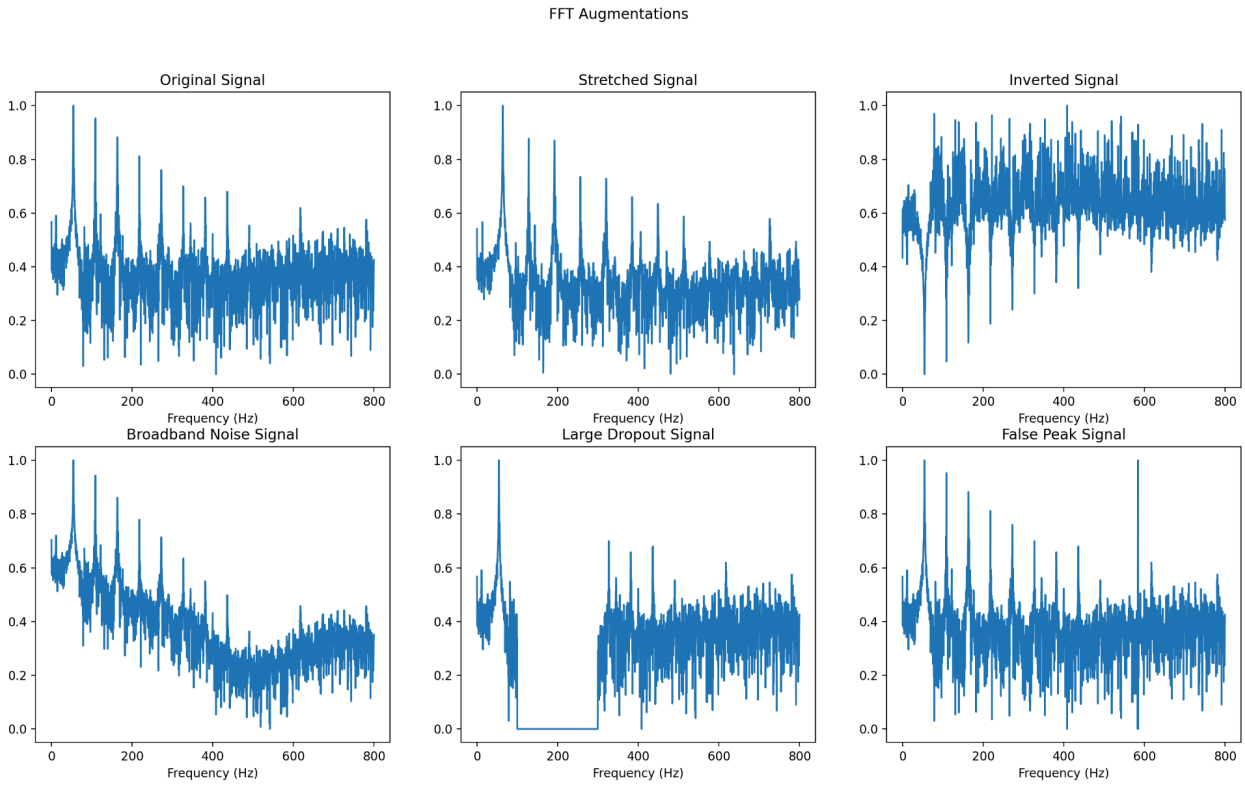



Figure 24: *Augmentations randomly applied every batch during training.*

# 7. Results

The table below provides a high level summary of the methods/approaches that were tried during this program, and what results were obtained as well as a ‘success’ indication that notes whether the method proved useful for the final system implementation. Following the summary table is a more detailed description of the noteworthy results.

| Approach | Result | Success |
|---|---|---|
| Dot Product | Comparing FFT vectors against an average undamaged vector did not result in reproducible ability to discriminate between types of damage on different days. | No |
| Time Domain | Training and testing on time domain data resulted in worse accuracy than frequency domain data. | No |
| FFT Subsection | No spectral subsection was found to work at the level of or better than the whole spectrum. | No |
| Other ML Architectures | No architecture of LSTM, Random Forest, and 2D Convolution worked better than the 1D convolutional network. | 1D Conv |
| Audiomentations | The only time domain augmentations that helped improve accuracy were gaussian noise, high pass filter, and low pass filter. More complicated time domain augmentations such as impulse response did not help. | Partial |
| Broadband Noise Augmentation | The augmentation of smoothed global changes in baseline frequency magnitude contributed the most of all the augmentations to the final accuracy. | Yes |
| Normalizing FFT | Both normalizing FFT blade frequencies and randomly altering FFT blade frequencies during augmentation improved final accuracy. | Yes |
| Averaging Augmentation | Taking two random input FFTs of the same damage state and averaging them together did not improve the final accuracy. | No |
| Repurposed Vision Augmentations | Adding false high values, cutting out large sections, and flipping the signal upside down all contributed minor improvements to the final accuracy. | Yes |
| Acceleration Direction as Color Visualization | Visualizing each direction X,Y, and Z, as colors R, G, and B respectively in a spectrogram, although allowing us to see that different frequencies were stronger in different directions, did not help us to identify defect signatures. | No |
| FFT Animation | Visualizing the change in FFT over time as a video of the 1D plot of the FFT over time allowed us to find an issue with our interpolation function. | Yes |
| GradCam Heatmap | Visualizing the resulting value magnitudes during inference within the neural network gave us some insight into what parts of the FFT the network was most using to make its calculations. Good for explainable AI. | Yes |

## 7.1. Dot product damage comparison

Initially, a global average undamaged feature vector was computed by subtracting the mean and

normalizing all undamaged FFTs. These normalized FFTs were then averaged together, resulting in a single "reference" undamaged FFT. Subsequently, the dot product between the reference vector and each individual day's FFTs corresponding to different damage states was calculated in order to identify any patterns or similarities. The vectors were normalized, and the averaging was performed over the pulse. The desired outcome was to observe similar groupings of dot products when multiplying by other undamaged FFTs compared to multiplying by the FFTs representing the damage states. However, it was found that while the dot products for undamaged vectors within data from the same day were closer to 1, this trend did not generalize across datasets collected from multiple days. The reason for this can be seen clearly in figure 16 which illustrates the large variability in undamaged vectors. The plot comparing the global average undamaged vector to the vectors of other states is shown below.

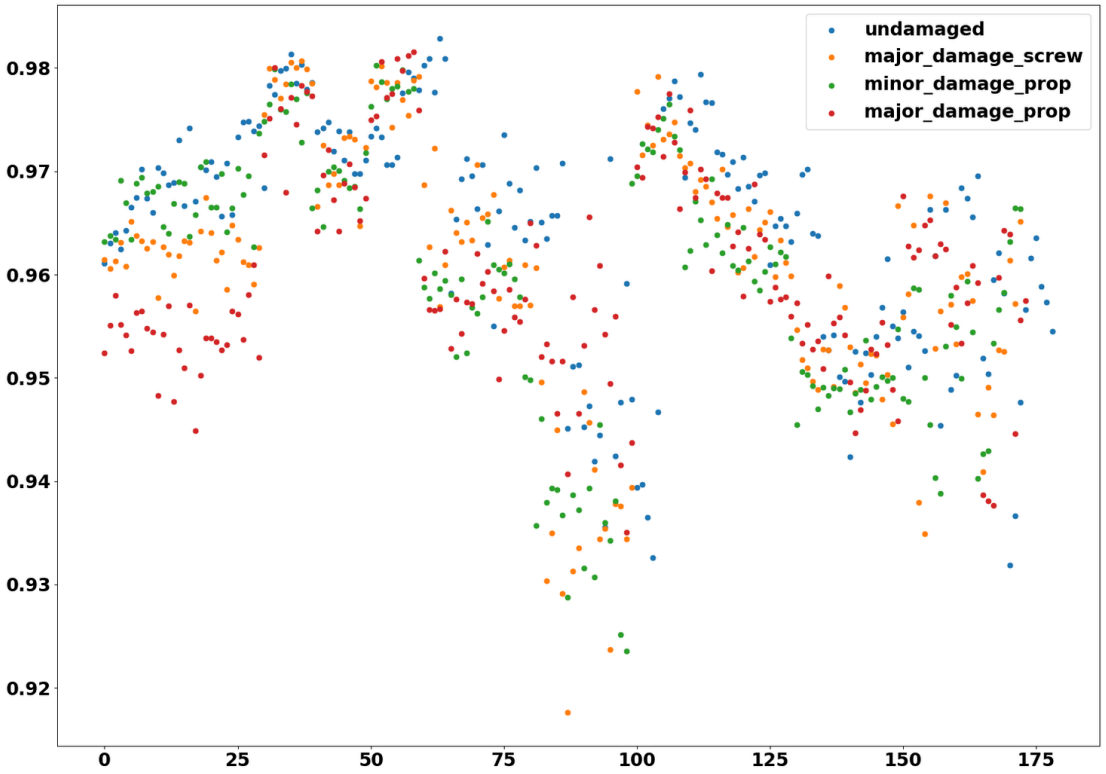



Figure 25: *Dot product of normalized FFT vectors with a normalized average undamaged vector. Each point represents the average dot product of a single pulse of vibration data.*

## 7.2. In-flight Results

The confusion matrix below is from an analysis of a single in-flight dataset created using the hover test procedure described in section 3.2. This dataset does not contain a blade major damage state because it was determined to be too dangerous to hover the vehicle stably in the small test chamber used with a damaged prop.

These results were generated using a version of the PreSound model from approximately halfway through the program, and may not be representative of the final performance obtained (see the end of section 7.4 for final detection results for in-flight data). They are still included here because the result importantly demonstrates the model's ability to differentiate damage states in-flight.

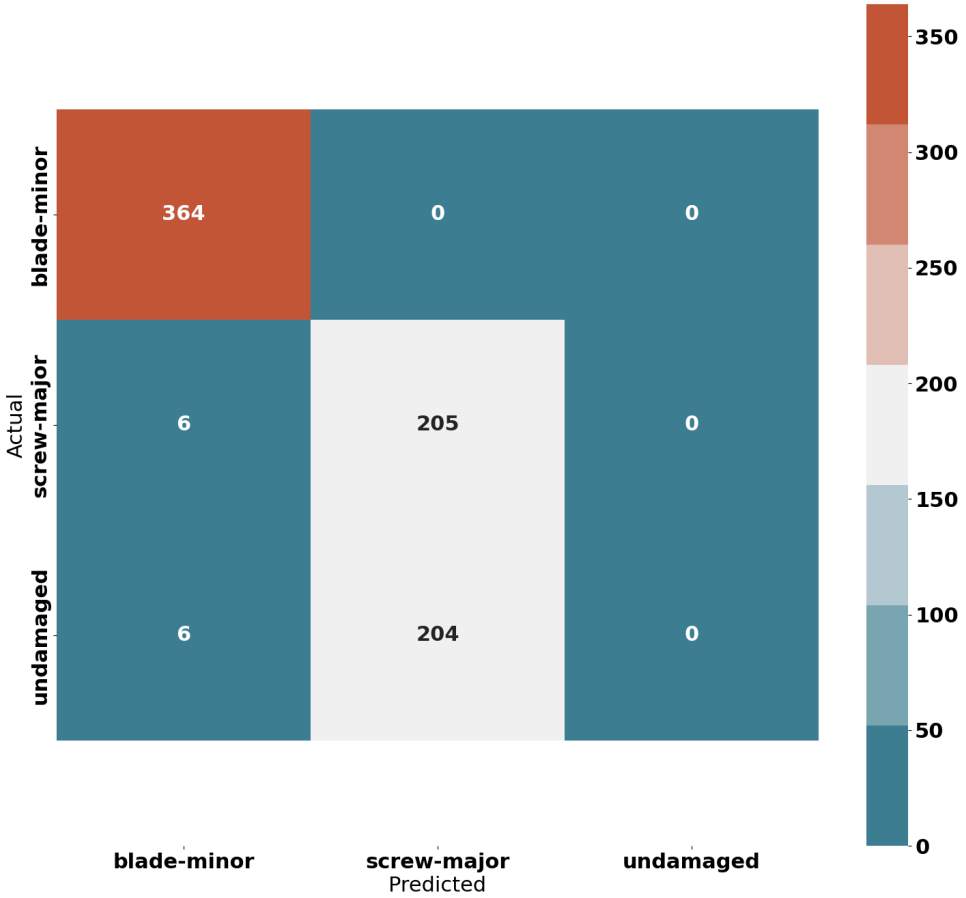



Figure 26: *Confusion matrix for in flight testing results.*

In Figure 26, the horizontal axis represents the category predicted by the final detection model, and the vertical axis represents the true category. Each count represents a short window of vibration data, which is processed via FFT to generate the prediction. For example the upper left square shows that 364 samples taken with blade-minor damage were classified as blade-minor, while the lower left square shows that 7 input samples with blade minor damage were incorrectly classified as undamaged.

This matrix shows that the blade damage was detected very consistently, with 364/376 blade minor damage samples being classified correctly. The loosene screw conversely was indistinguishable from no damage in the in-flight data, with only 50% classified correctly (i.e. no better than a random guess).

### 7.2.1. Presound-L Vs On-Board Comparison

During the initial flight testing, IMU data was captured from both the on-board vehicle IMUs and the externally attached Presound-L IMU (described in section 3.2). This allowed for a direct comparison to validate the proper functioning of the Presound-L node and to assess whether the mounting interface introduced significant damping or interference to the Presound-L sensor.

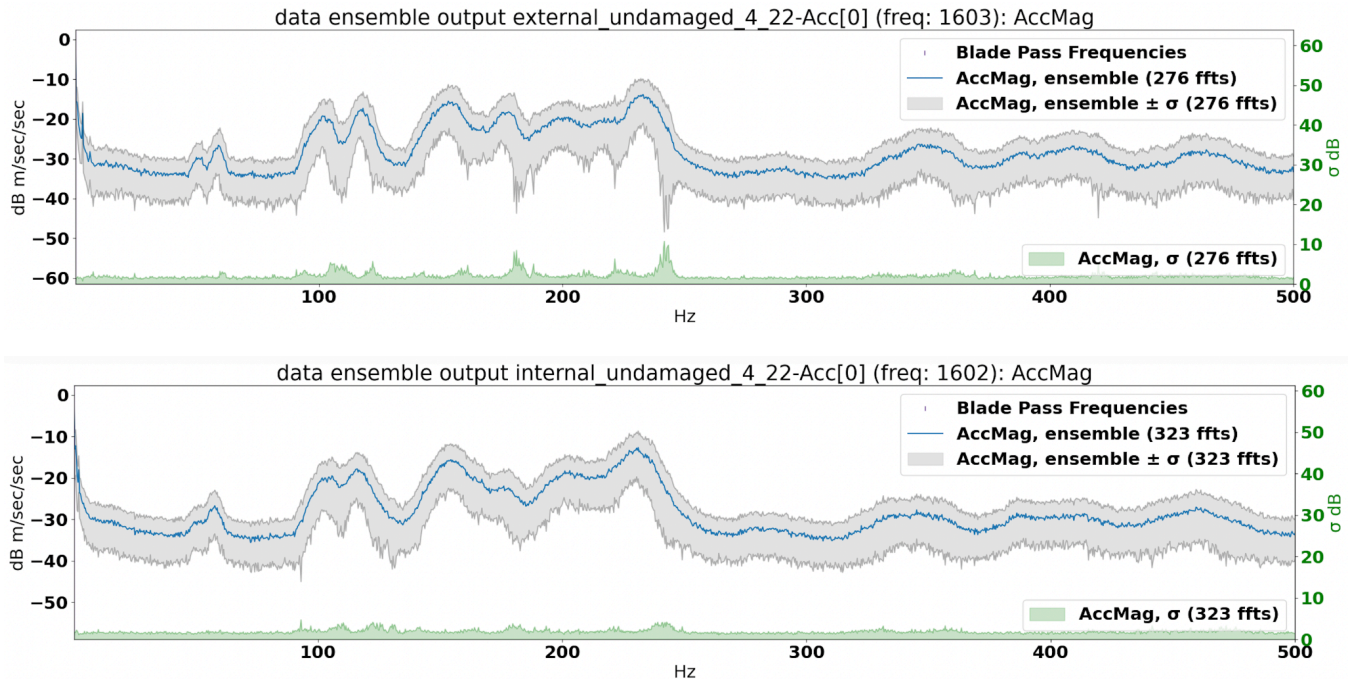



Figure 27: *Comparison of on-board IMU ensemble average FFT (top) to externally attached presound-L ensemble average FFT (bottom) measuring the same indoor hover test. Note: the blade pass frequency lines are not displayed due to varying motor RPMs in flight.*

### 7.3. FFT Subsection Testing

To gain a better understanding of the FFT sections that were most significant in detecting the damage state, the machine learning algorithm was executed with all but one quarter of the input set to zero in the frequency domain. Note that the data used for this analysis were those from before the increase in throttle. The results revealed that the 400-600 Hz range contained the most relevant information for determining the damage state for the on-ground data using the 25% throttle state. Nevertheless, employing the full spectrum for classification purposes still yielded superior results compared to using any isolated subsection alone.

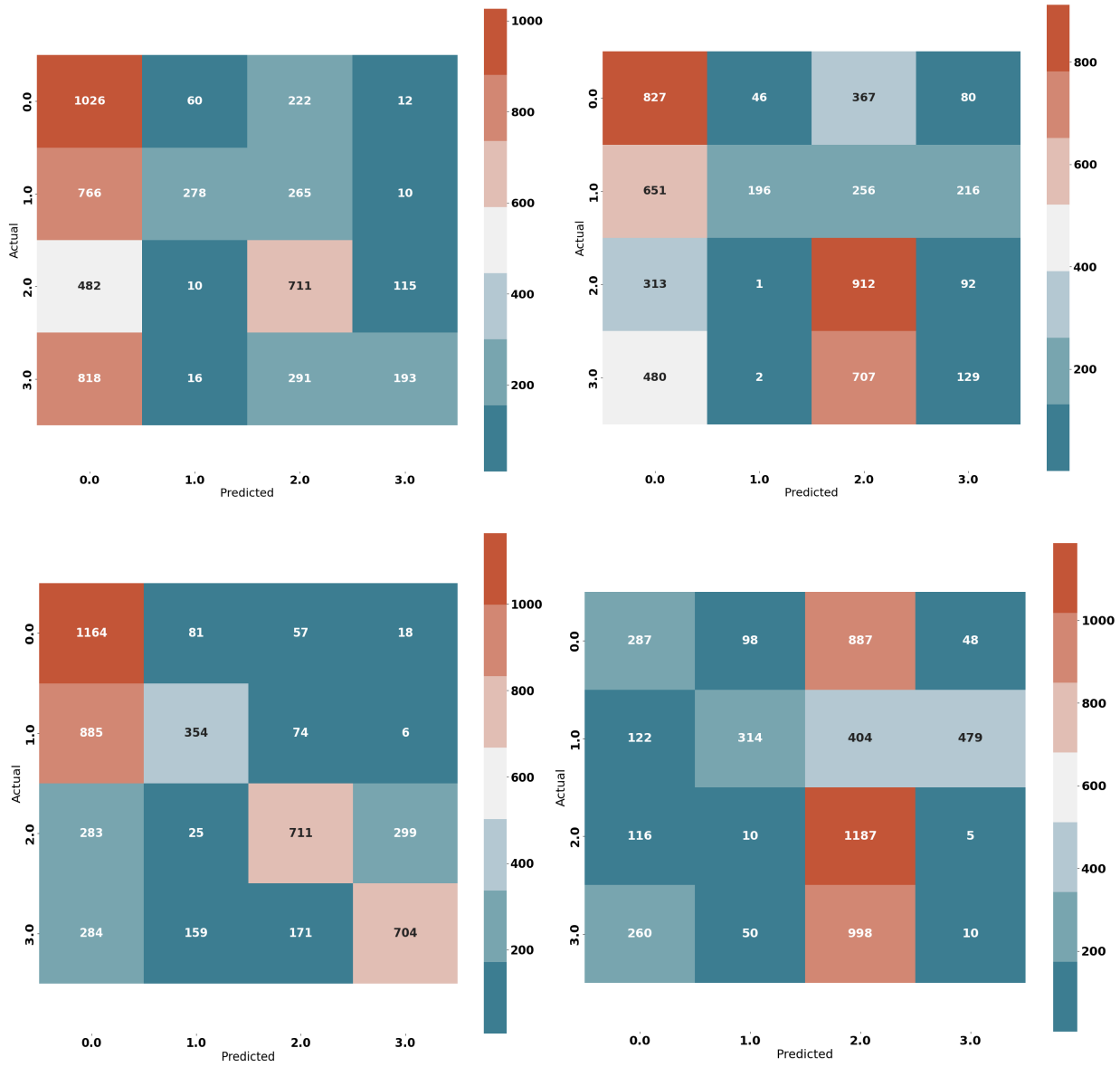



Figure 28: *A set of confusion matrices of the count of predictions for each damage state of the low throttle data on 0-200 Hz (top left), 200-400 Hz (top right), 400-600 Hz (bottom left), and 600-800 Hz (bottom right). Labels 0,1,2,3 correspond to undamaged, screw major damage, blade minor damage, and blade major damage respectively.*

Performing this same test on the data collected with 50% throttle showed very different results. In this case, the 0-200 Hz range contained the most relevant information for determining the damage state. The test accuracy of this range, 99%, was as good or better than the results from the full spectrum (see section 6.4). The accuracy dropped to 91% for the 200-400 Hz range, then down again to 44% and 54% accuracy respectively for the 400-600 Hz and 600-800 Hz ranges. This suggests that when the throttle level is increased, highly relevant vibration data begins to show in lower frequency ranges, with much less

relevant data in the higher frequency ranges. See section 6.5 for further investigation into this phenomena.

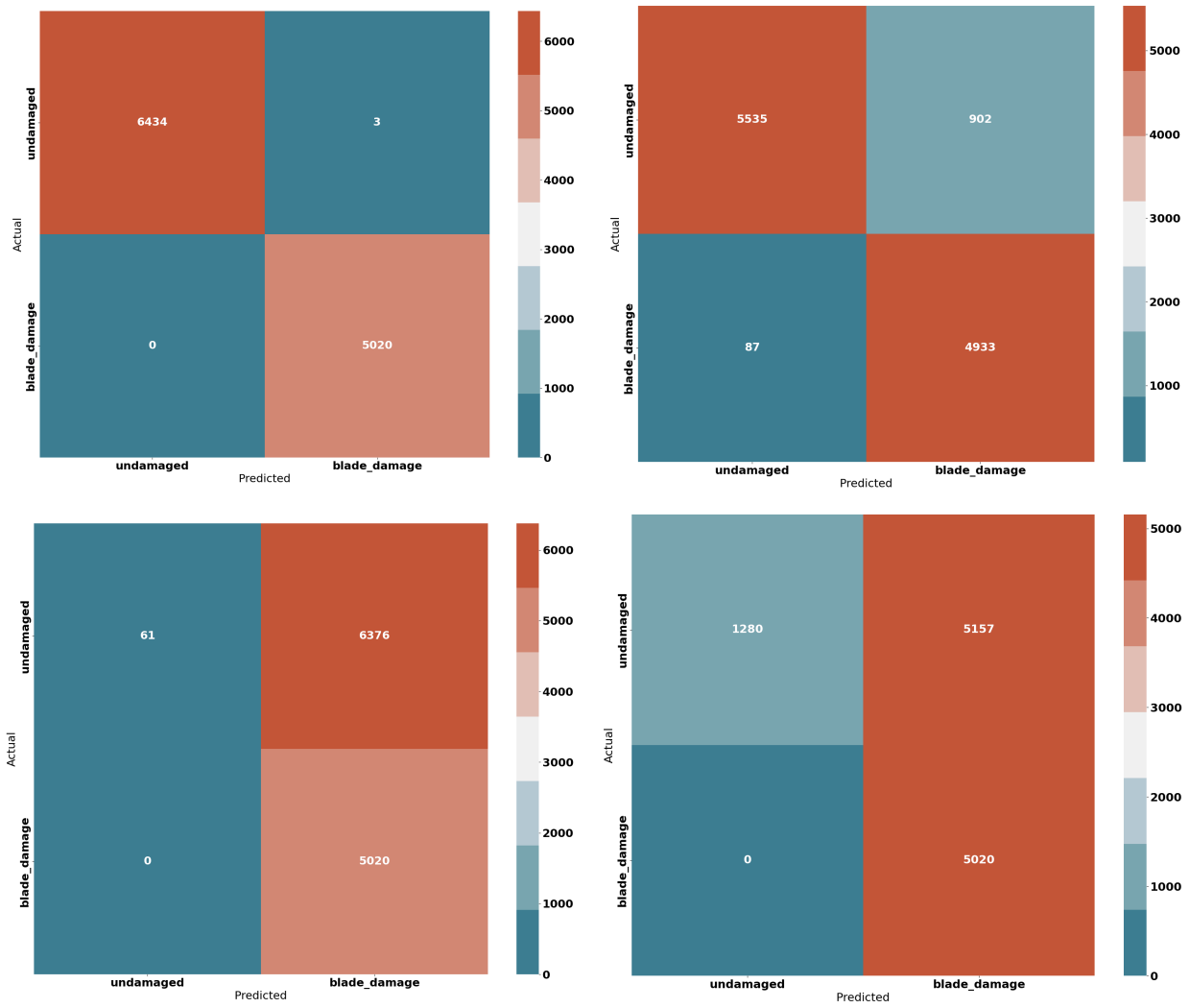



Figure 29: *A set of confusion matrices of the count of predictions for each damage state of the high throttle data on 0-200 Hz (top left), 200-400 Hz (top right), 400-600 Hz (bottom left), and 600-800 Hz (bottom right).*

## 7.4. Final Detection Results

The final detection model, trained on data captured with props spinning at 50% throttle from 22 capture files containing 395 pulses in total, demonstrates excellent performance, achieving an average accuracy of over 99% for detecting blade damage on a single FFT vector input when all augmentations are applied. Furthermore, the trained network exhibits the ability to generalize to new types of blade damage. When tested on a different type of blade damage than the one it was trained on, the new damage case was correctly classified with 98% accuracy. Since each pulse consists of a large set of input FFT vectors, this means that test pulses overall are classified with 100% accuracy. Finally, when this model was tested on-vehicle under live conditions, all data captures (each being 10s of vibration data) were classified

accurately with over 95% confidence, meaning that at least 95% of the FFTs within each capture were classified correctly.

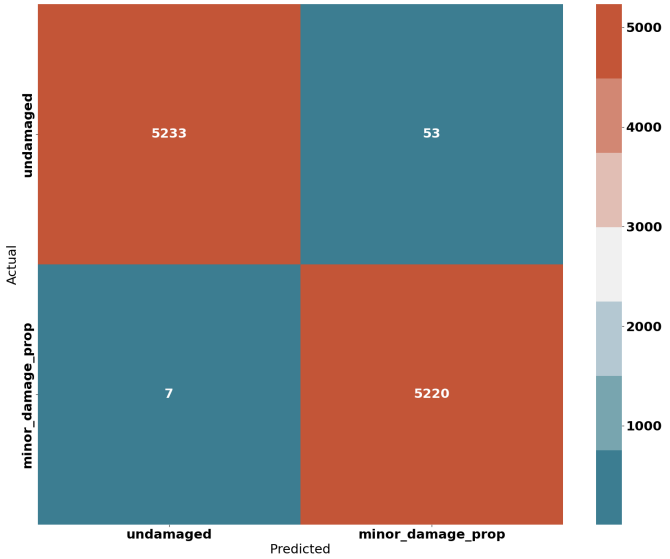



Figure 30: *Confusion matrix showing rate of correctly identifying minor damage vs. no damage.*

This notable improvement in performance over what was achieved earlier in the program can be largely attributed to the increase in throttle during training. However, the augmentations identified when training on data from a lower throttle setting also contribute to enhancing the network's generalizability. Without these augmentations, accuracy on a separate day decreases from 99% to 97%. This accuracy is high enough to indicate that more subtle damage states should also be detectable with a similar method.

Figure 31: *Examples of blade minor (left) and blade major (right) damage.*

When the model is trained and tested on in-flight data, it achieves an average accuracy of 85% in discriminating between undamaged and blade-damaged states across 10 tests, with a peak accuracy of 91%.

### 7.5. GradCam Visualizations

GradCAM, short for Gradient-weighted Class Activation Mapping, is a technique commonly employed in computer vision for 2D convolutional networks. It capitalizes on the fact that convolutions possess a

natural mapping that corresponds to the input but at a reduced scale. By weighting the activations of the final convolutional layer with gradient information, GradCAM generates a coarse localization map. This map provides insights into the regions of the input that the network considers significant in distinguishing between different damage states.

In our analysis, we applied GradCAM to investigate the decision-making process of the machine learning model in relation to determining damage from the Fast Fourier Transforms (FFTs). The results demonstrated that the model predominantly focuses on frequencies immediately preceding the RPM peak, particularly within the lower frequency range. By emphasizing these specific frequency components, the model effectively discriminates between the different damage states under examination.

Test set:

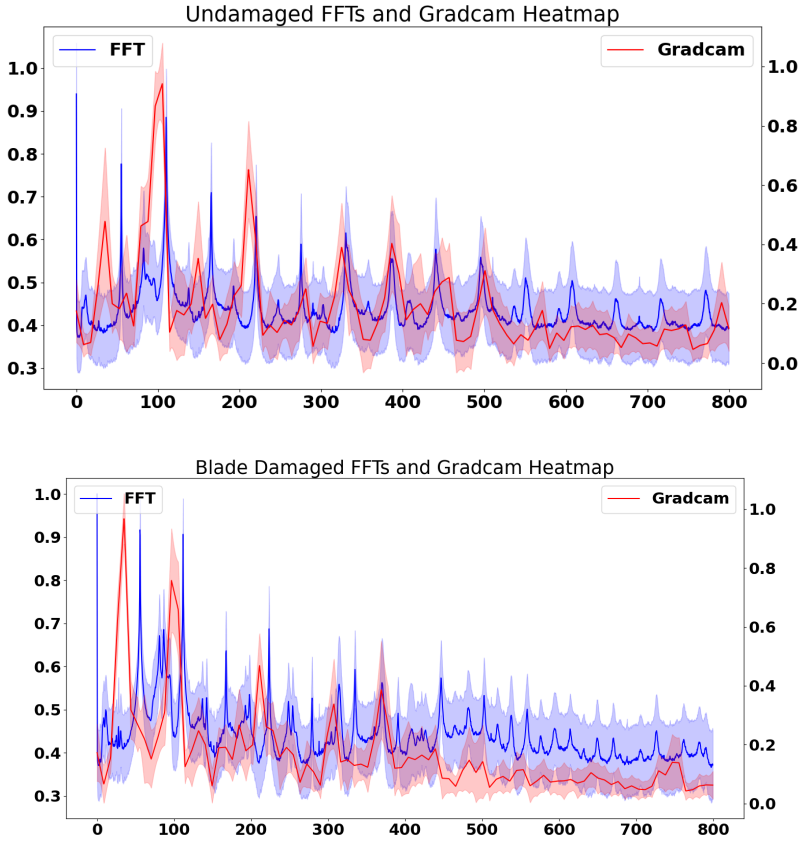



Figure 32: *Average normalized GradCam heatmaps in red overlaying average undamaged (top) and damaged (bottom) FFT signals in blue for the separated test set of vibration pulses.*

Training set:

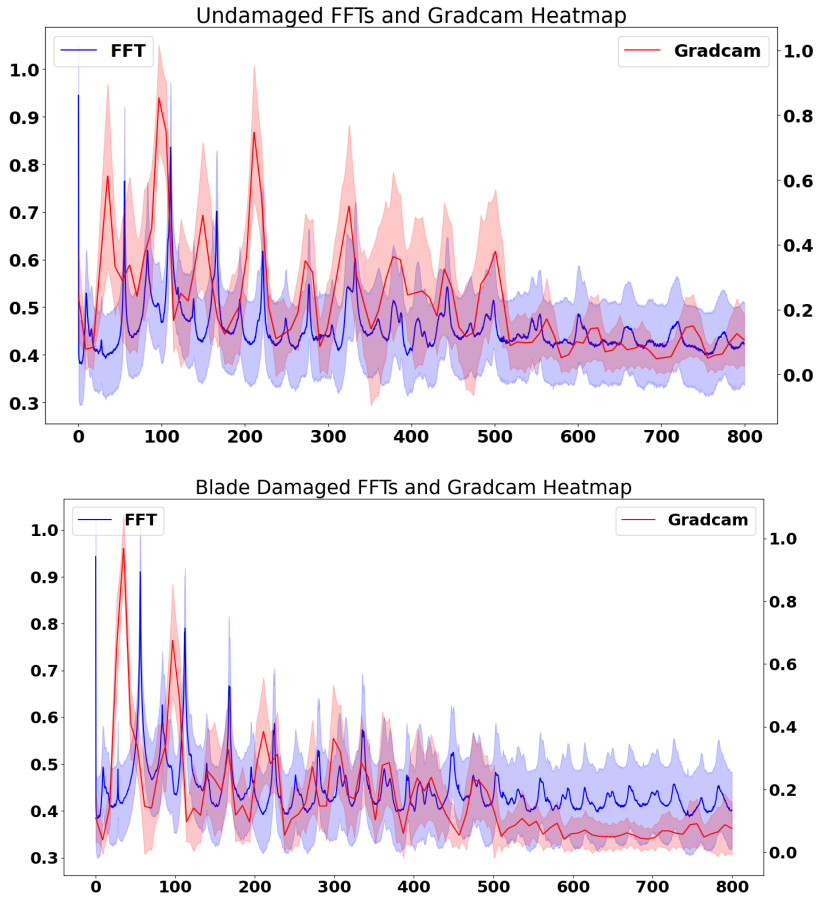



Figure 33: *Average normalized GradCam heatmaps in red overlaying average undamaged (top) and damaged (bottom) FFT signals in blue for the vibration pulses that the network trained on.*

# 8. Discussion

## 8.1. Model Generalizability

The single largest hurdle encountered during this project arose approximately one year in when conducting model robustness testing. Unexpectedly, the accuracy of the results was much lower than before, deviating from the previously consistent pattern. Prior to this point, large volumes of vibration data were typically collected and then randomly divided into train/test/validation sets for evaluating the model. The test data had not been seen by the model during training, but it originated from the same comprehensive dataset. This setup yielded consistently high accuracy in most tests.

The introduction of a new testing method, designed to assess model generalizability, unveiled the issue. Instead of randomly separating the test set from the larger dataset, independent full data gathering sessions were dedicated solely to testing. For example, we trained the model using data exclusively from day 1 and evaluated it on data solely from day 2. With this change, the average accuracy on the testing sets significantly dropped. This indicated a few potential factors:

1. Substantial background variations occurred between data gathering sessions, making it necessary for the model to have a larger dataset to effectively interpret all these variations as belonging to the same damage class.
2. The data contained subtle markers, distinct from the actual damage signal, that were unique to

each data gathering session. The model may have relied on these markers to predict the damage state. For instance, if a constant background tone (e.g., from an air conditioning system) was present in all blade damage data collected on a particular date, and the same tone appeared in undamaged baseline data gathered on a different date but at a different frequency, the model might have learned to associate the frequency of this background tone with the damage state rather than identifying the genuine damage signal.

Upon making this discovery, GreenSight prioritized an extensive investigation into the cause. This investigation led to the identification and resolution or improvement of various issues within our data gathering and processing pipelines. These enhancements were implemented to ensure the model could maintain high accuracy even when applied to data collected on entirely new days. The subsequent sections describe the components that were investigated and the improvements that were made.

### 8.1.1. Data Splitting System

The data splitting system employed in the initial phase involved dividing each pulse into three sections. Although this method demonstrated generalizability from one FFT to another at the same time, it failed to assess the system's ability to generalize to different pulses or data gathering sessions. To address this limitation, modifications were made to the data splitting process. A dimension was added to the input data to enable manipulation at the pulse level, allowing for randomized splitting of pulses into the various datasets. This revealed slight generalizability issues, and further cross-validation efforts highlighted a wide range of possible accuracies on test data. Notably, if pulses from a specific day were randomly assigned to the test set, the model lost its ability to classify that data. To overcome this challenge, a refined data splitting system was developed, enabling the separation of bin files from specific days. The code was adjusted to allow users to specify which days should be allocated to the test set. This approach facilitated a comprehensive assessment of the model's generalizability during and after training sessions.

### 8.1.2. Model Structure

The model's architecture itself was another factor that could impact its ability to generalize effectively. To investigate this, multiple alterations were tested. For PreSound, a one-dimensional convolutional network was chosen due to its efficiency in capturing repeated patterns in spatial data, which was evident in the harmonic structure of the vibration data caused by blade rotation. Since the input was based on the one-dimensional FFT signal, 1D convolutions were preferred over 2D convolutions. Although the accelerometer had four channels (X, Y, Z, and Magnitude), this number is insufficient to capture repeating patterns, even if additional channels were created to represent relationships between the accelerometer channels. The 2D convolutional network was still evaluated but did not yield improved performance compared to the smaller 1D model. Increasing the number of convolutional layers did not enhance performance either, leading to the decision to enlarge the kernel size to cover a wider range of the frequency spectrum. Dropout layers within the convolutional section were replaced with 1D spatial dropout layers, ensuring consistent dropout spots across all channels. These architectural adjustments resulted in minor improvements in generalizability and performance compared to the original architecture.

### 8.1.3. Physical Setup

Variations in the physical testing setup were identified as potential factors impacting model generalizability. For instance, differences in vehicle battery levels or variations in how the vehicle was mounted in the test cradle could have introduced discrepancies in the collected vibration data that the model struggled to interpret without sufficient training examples accounting for such variations. To investigate this, a series of experiments were conducted to control for different physical variables, including:

- Vehicle battery level
- Vehicle mounting position in test cradle (i.e. rotating the vehicle to different orientations in the cradle)
- Power source (battery alone or battery with wall power connected)

- Effects of vehicle contact points with the cradle (isolated by using foam padding or no padding)
- Motor RPM

While it was expected that some of these conditions would lead to reduced accuracy or necessitate additional data to account for the variations in the model, most of these parameters did not significantly affect the overall accuracy or generalizability. The exception was motor RPM, where an increase from 25% to 50% throttle (corresponding roughly to an RPM change of 2000 to 3000) had a noticeable impact on the model's ability to generalize.

Additionally, although it did not directly affect the model's accuracy or generalizability, the incorporation of foam padding to eliminate potential "rattle" effects during data collection led to more consistent data, reducing the amount of training data required to build an accurate model. As a result, foam padding became a standard part of the testing configuration.

### 8.2. Sample Rate During Streaming

Initially, the performance of the system significantly declined when employing log streaming for sensor data. We hypothesize that the primary factor contributing to this outcome is the sample rate reduction from 1600 Hz to 1000 Hz, which is inherent in the configuration of the streaming process. However, it should be noted that this sample rate drop alone does not account for all the observed variations in the results. The frequency spectra derived from the live streamed 1000 Hz data do not align well with the spectra obtained by downsampling the 1600 Hz data to 1000 Hz, as would be expected.

It is plausible that the downsampling method preserves certain high-frequency components more effectively than direct sampling at the lower rate. This could be attributed to the difference in the uniformity of sampling times, as described in Section 4.2 of this report. However, despite the initial challenges, the final model exhibited the ability to undergo rapid retraining using streaming data for live testing. Moreover, when deployed for live inference on a Jetson Nano platform [5], consistent and high-quality results were achieved. This achievement could be attributed to the observation that our final model relies predominantly on lower-frequency patterns for making accurate decisions, as discussed in the subsequent section.

### 8.3. GradCam Analysis

In general, the GradCam heat maps closely resemble the underlying shape of the FFT. Both heatmaps primarily emphasize frequencies just before each peak, especially in the lower range. Specifically, the heatmap for blade damage exhibits a stronger sensitivity to low frequencies, particularly around ~50Hz, while largely disregarding frequencies above 500Hz compared to the undamaged heatmap.

Interestingly, the heatmaps for the test set disregard several peaks that are not ignored in the training set. For instance, the peak between 100-200Hz is virtually eliminated. This observation suggests that these specific parts of the spectrum may contain markers that do not generalize between sets. As a result, they could be potential candidates for suppression or blurring during training if they exhibit consistency. This approach would ensure that the model does not rely on these markers exclusively.

## 9. Summary

From 2021-2023 GreenSight undertook a research project to develop and evaluate a method for monitoring aerial vehicle structural and propulsion health using vibration data. Datasets were collected using GreenSight's small quadrotor UAS as a test vehicle, both by spinning the vehicle's props in a test pattern on the ground and by hovering the vehicle in the air.

Two sources of vibration data were used; accelerometers from the vehicle's built-in flight controller, and accelerometers externally mounted to the vehicle's airframe. Built-in accelerometer data was sufficient for

fault detection on the small test UAS, while it is expected that the externally mounted sensors will be required for larger aircraft.

A machine learning model was trained to identify various cases of damage to one of the vehicle's props or a loose screw in the airframe. Once trained this model could run directly on the vehicle, predicting damage state every 10s.

Initially this model performed poorly when tested over long time periods, indicating an inability to generalize well. This deficiency was eventually corrected, with the throttle setting of the motor emerging as a key factor impacting the performance of the damage signal detection pipeline. Specifically, increasing the motor throttle to induce stronger excitation was the most influential modification to the pipeline that yielded improvements in the overall detection outcomes.

Final results showed that the model was able to reliably detect prop damage during on ground testing, able to correctly differentiate between damaged and undamaged props 30/30 times during live demonstration, and achieving 98% average accuracy when classifying individual FFT spectrums randomly selected from more than 40 sets of captured vibration data. The loose screw damage state was not reliably detected, with the model achieving barely better than 50% accuracy. In-flight results achieved 85% average accuracy for prop damage detection (for individual FFTs), but the amount of tests run is much smaller for in-flight so these results are preliminary when compared to the on-ground results.

A critical consideration in the development of machine learning systems for vibration analysis is the appropriate allocation of data for training, validation, and testing purposes. The findings from this project underscore the presence of significant variations in signal patterns across different days of data collection, even when rigorous efforts were undertaken to maintain consistent experimental conditions. To mitigate the potential introduction of dataset bias, it is recommended to reserve entire days of data collection exclusively for testing purposes.

Several additional factors warrant consideration for future vibration analysis projects, including disparities in sample rates during data streaming, the incorporation of increased data variability into the training set, and the suitability of the machine learning model architecture for the specific characteristics of the data.

## 10. References


1. *Feng, K., Ji, J. C., Ni, Q., & Beer, M. (2023). A review of vibration-based gear wear monitoring and prediction techniques. Mechanical Systems and Signal Processing, 182, 109605. https://doi.org/10.1016/j.ymssp.2022.109605*

2. *Eltouny, K., Gomaa, M., & Liang, X. (2023). Unsupervised Learning Methods for Data-Driven Vibration-Based Structural Health Monitoring: A Review. Sensors, 23(6), 3290. https://doi.org/10.3390/s23063290*

3. *Abdeljaber, O., Avci, O., Kiranyaz, S., Gabbouj, M., & Inman, D. J. (2017). Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. Journal of Sound and Vibration, 388, 154–170. https://doi.org/10.1016/j.jsv.2016.10.043*

4. *Malekloo, A., Ozer, E., AlHamaydeh, M., & Girolami, M. (2022). Machine learning and structural health monitoring overview with emerging technology and high-dimensional data source highlights. Structural Health Monitoring, 21(4), 1906–1955. https://doi.org/10.1177/14759217211036880*

5. *Cass, S. (2020). Nvidia makes it easy to embed AI: The Jetson nano packs a lot of machine-learning power into DIY projects - [Hands on]. IEEE Spectrum, 57(7), 14–16. https://doi.org/10.1109/MSPEC.2020.9126102*

6. *AD Team. (2016). Ardupilot. URL: www.ardupilot.org/*

7. *Meier, L. (2016). Pixhawk autopilot. URL: https://pixhawk.org.*

# 11. Appendices

## 11.1. Variable Sampling Rate Study

In Phase I external sensors, capable of uniform sampling, were used. However, the project scope in Phase II favored onboard sensors which have some limitations in terms of uniform sampling, mostly due to the onboard flight controller limited resources and logging device memory and buffer size. Figure 34 shows sampling rate variation for a sample data file. Therefore, we investigated the performances of different non uniform techniques for taking the FFT of the signal. More specifically, we tried (i) interpolation techniques including linear interpolation, Fourier transform resampling, and sinc interpolation for generating uniform samples, and (ii) directly taking FFT from the non uniform samples with a function called `NUFFT` (non uniform FFT) implemented in MATLAB. We benchmarked these techniques using synthesized sinusoids and previous data collected in Phase I.

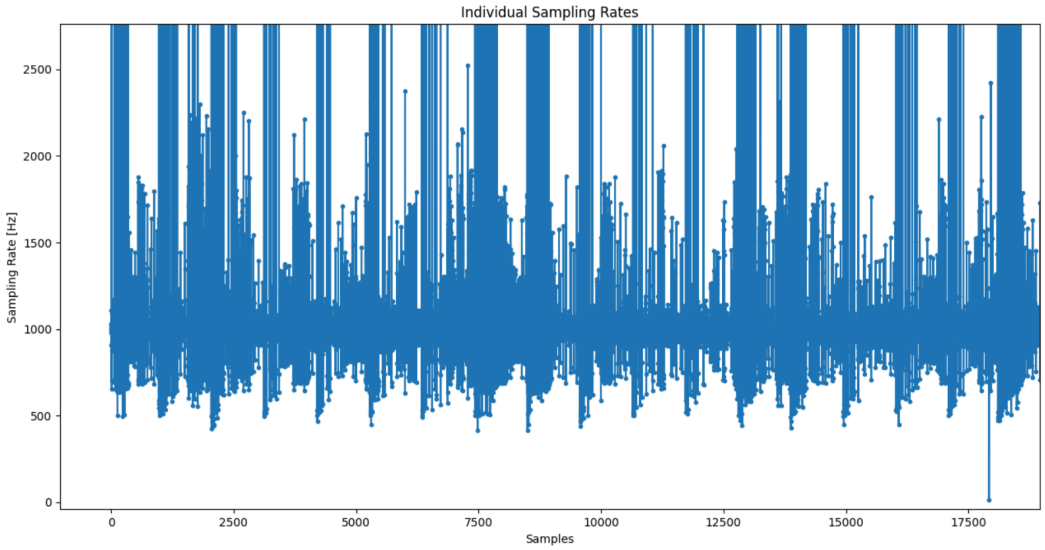



Figure 34: *Sampling rate variation of a sample data file.*

For each case (sinusoids or previous data) we randomly decimated (downsampled) the signal. Then, we interpolated the data to generate data with uniform sampling time and took FFTs from the interpolated data. Moreover, we took NUFFTs from the decimated data and we used the FFT of the original data as the benchmark. Details of the interpolation techniques are as follows.

Linear interpolation is found by the concatenation of linear interpolants between each pair of data points in the decimated signal. It is the fastest technique amongst the others and implemented by `interp1` function in MATLAB and it has different implementations in Python.

Fourier transform method is implemented by `scipy.signal.resample`[1] in Python, and uses the FFT to find the Fourier coefficients for generating the resampled data. One can say it is not efficient to first take FFT for generating resampled (interpolated) data and take another FFT from that data. Nonetheless, we did it to have a thorough comparison on resampling techniques.

The Whittaker–Shannon interpolation formula or sinc interpolation is a method to construct a continuous-time bandlimited function from a sequence of real numbers[2]. Based on sinc interpolation

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.resample.html

[2] https://en.wikipedia.org/wiki/Whittaker%E2%80%93Shannon_interpolation_formula

method, given a sequence of real numbers, x[n], the continuous function used as the generator for the resampling can be written as

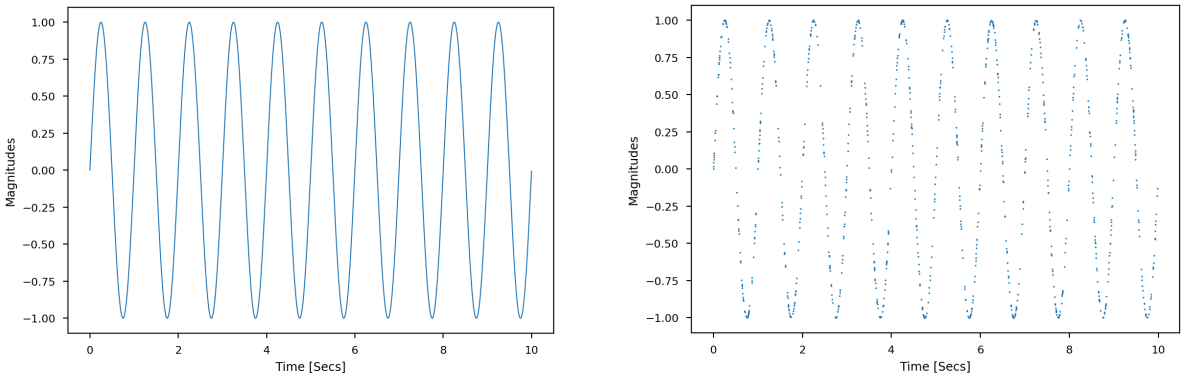
$$x(t) = \sum_{n=-\infty}^{\infty} x[n] sinc(\frac{t-nT}{T}),$$

where sinc(.) denotes the normalized sinc function and 1/(2T) is the desired band limit in hertz. We implemented the sinc function ourselves. The $nufft$ is implemented in MATLAB based on Nonuniform Discrete Fourier Transform of Vector[3]. The nonuniform discrete Fourier transform of a vector X of length n, sample points t, and frequencies f is defined as

$$Y(k) = \sum_{j=1}^{n} X(j) e^{-2\pi i\, t(j) f(k)},$$

where $k = 1, 2, ..., m$. When $t = 0, 1, ..., n - 1$and $f = \frac{0, 1, ..., n-1}{n}$ (defaults for $nufft$), the formula is equivalent to the uniform discrete Fourier transform used by the $fft$ function.

Figure 35(a) illustrates a time-based sample of synthetic data considering $x = sin(2\pi t)$. Non-uniform samples were generated by decimating the data, as depicted in Figure 35(b). The results of Fast Fourier Transform (FFT) for the various mentioned techniques were then compared with the FFT of the original data. Interpolation techniques were applied to the decimated data, as shown in Figure 36, while Figure 37 displays the FFT of the interpolated data.




(a) Original signal (b) Decimated signal with keep ratio of 0.1

Figure 35: *Synthetic sample data with (a) uniform and (b) nonuniform sampling.*

[3] https://www.mathworks.com/help/matlab/ref/double.nufft.html

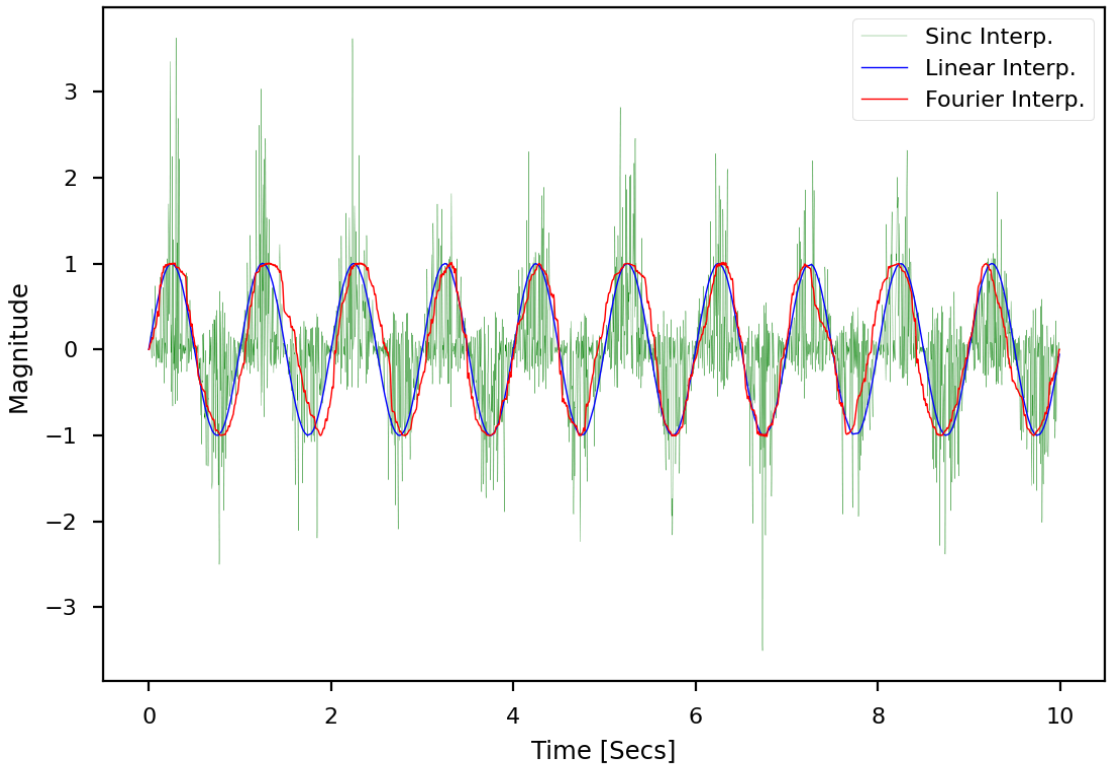



Figure 36: *Uniform sampling using different interpolation techniques.*

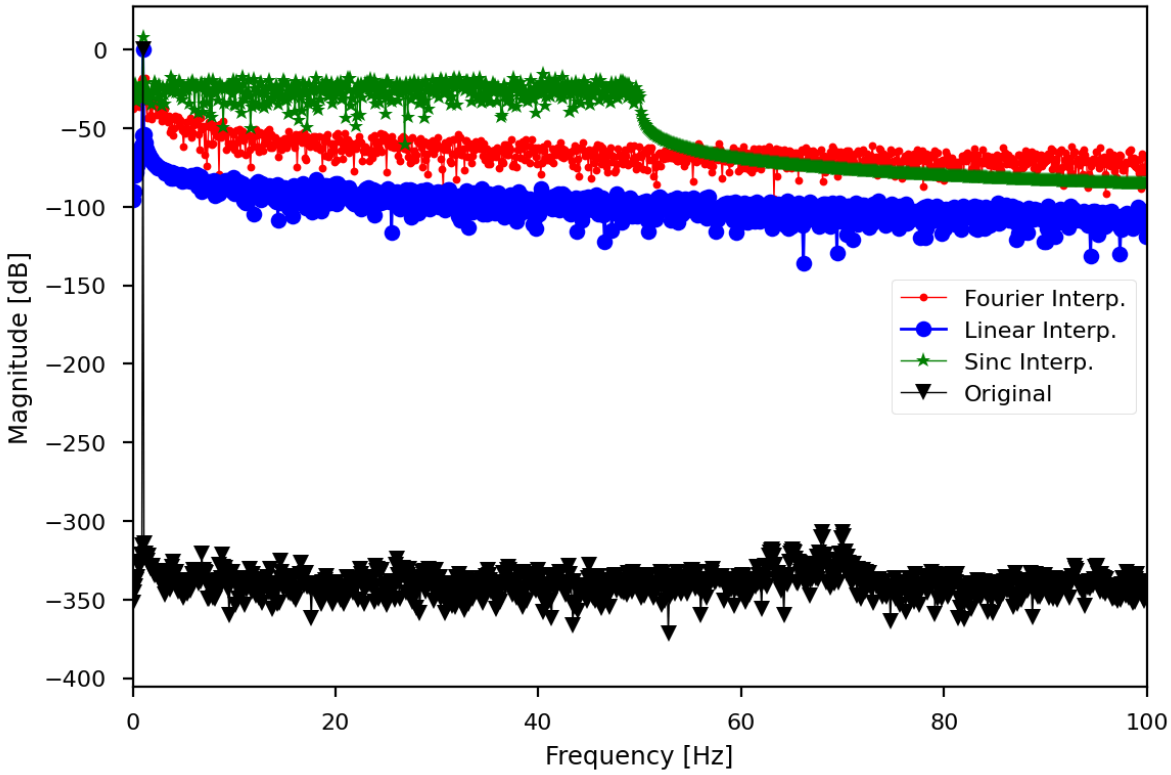
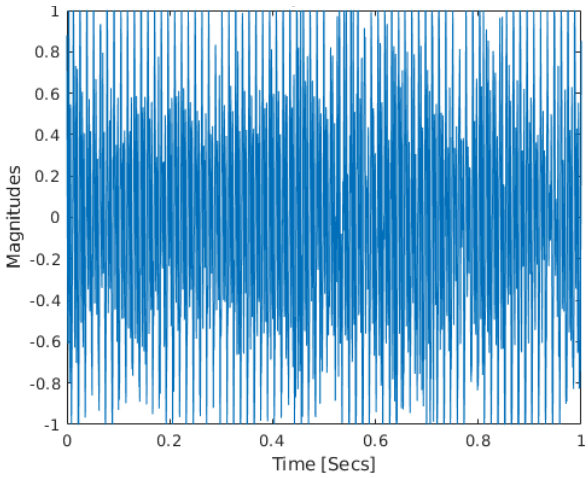



Figure 37: *Comparison of the FFT of the interpolated data and the original data*

In Figure 38, a sample signal from the IMU, ACC_X, is shown over time. The decimation process was applied to this signal. The decimated data were then analyzed and the analysis compared to the analysis of the original signal. Figure 39(a) displays the FFT after the decimated signal was uniformly resampled using linear interpolation. Figure 39(b) shows the FFT of the original data. Figure 40 shows the FFTs on the same plot. This demonstrates that linear interpolation of unequally sampled data can be used to obtain a reliable FFT.




FFigure 38: *IMU data*

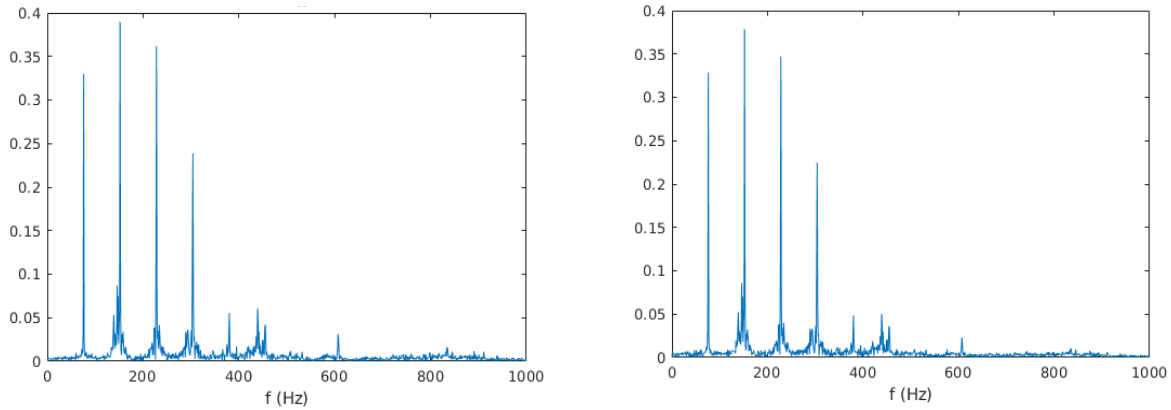



Figure 39: *FTT of the (a) original and (b) resampled decimated data. The vertical axis shows the FFT absolute value.*

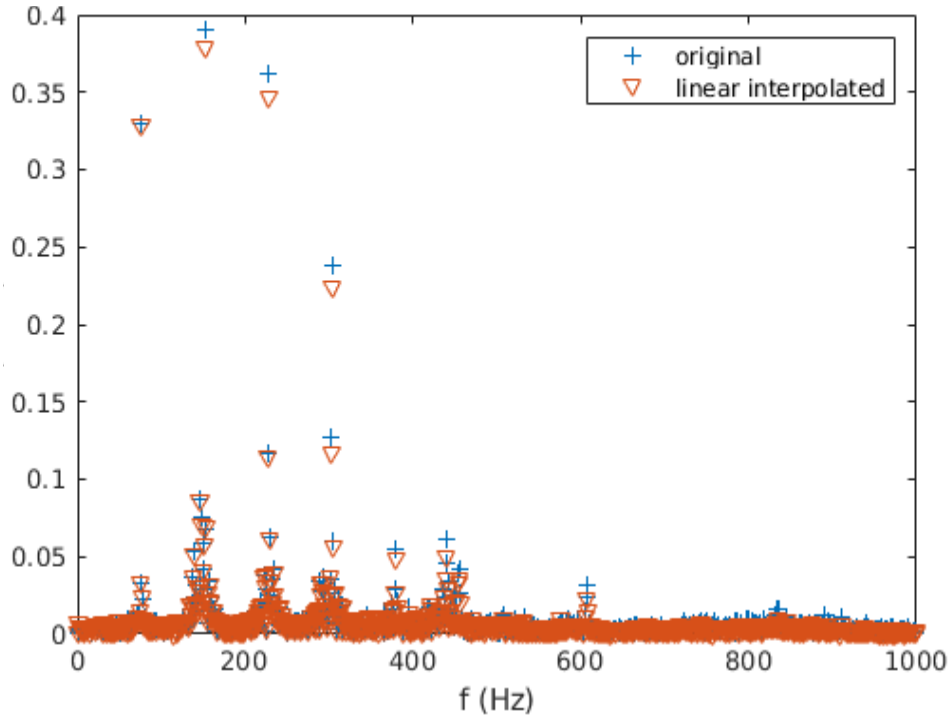
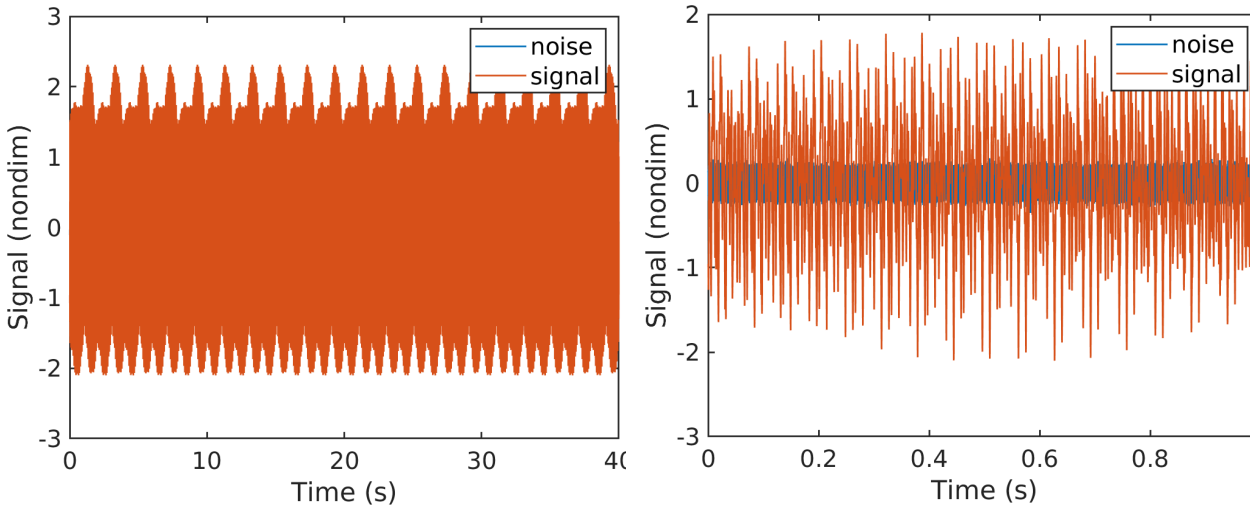



Figure 40: *Comparison of the FFT of the interpolated decimated and the original IMU data. The vertical axis shows the FFT absolute value.*

Subsequently, the FFT of the uniform data resampled via linear interpolation is compared with the Non-Uniform Fast Fourier Transform (NUFFT) of the non-uniform data. A signal synthesized by summing five sinusoids at the peak frequencies from a sample IMU data is employed, with the inclusion of added noise. Once again, the data is decimated to create non-uniformity. Since the exact FFT of the synthesized data is known, it provides a better understanding of the performance of the Linear-Interpolation (LI-FFT) and NUFFT methods. Figure 41 displays the synthesized signal with the five sinusoids, while Figure 42 shows the numerical and exact analytical FFT of the original signal. Figure 43 illustrates the LI-FFT and NUFFT of the decimated data along with the exact FFT of the original data. Furthermore, Figure 44 compares the LI-FFT and NUFFT for two different keep ratios of the decimated data, using the normalized error as a measure. The normalized error represents the absolute difference between the exact FFT and numerical FFT, normalized with respect to the exact FFT.




(b) Original signal

(b) Closer look at the original signal

Figure 41: *Synthetic sample data with 5 sinusoids.*

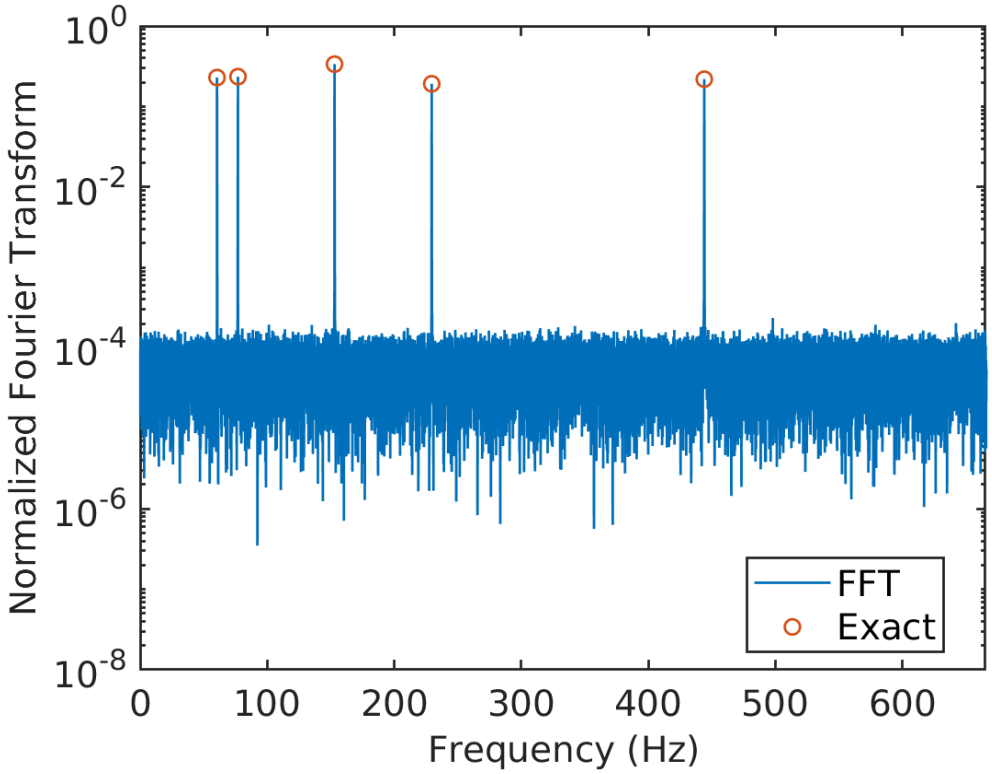
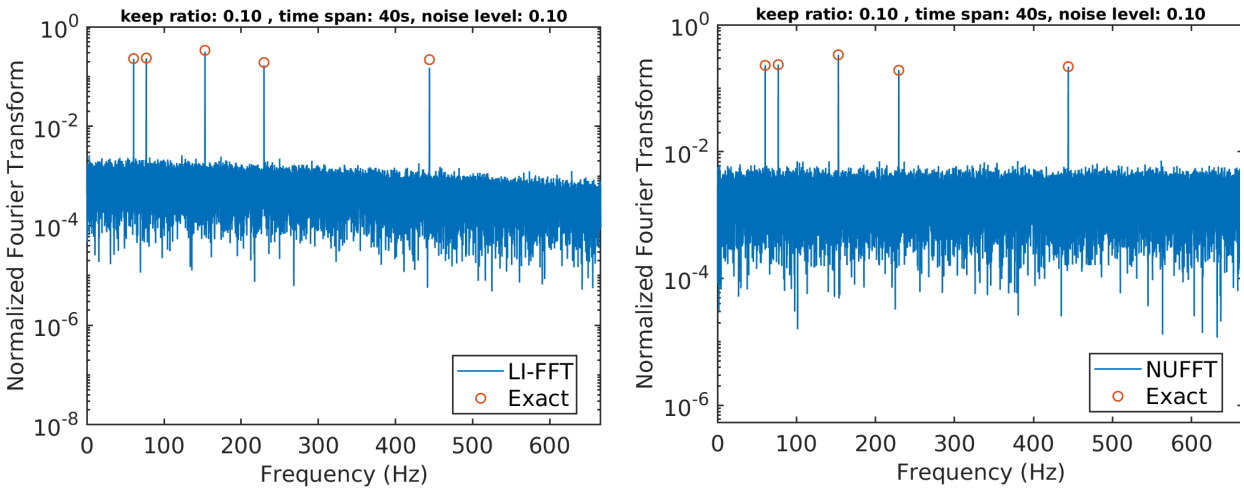
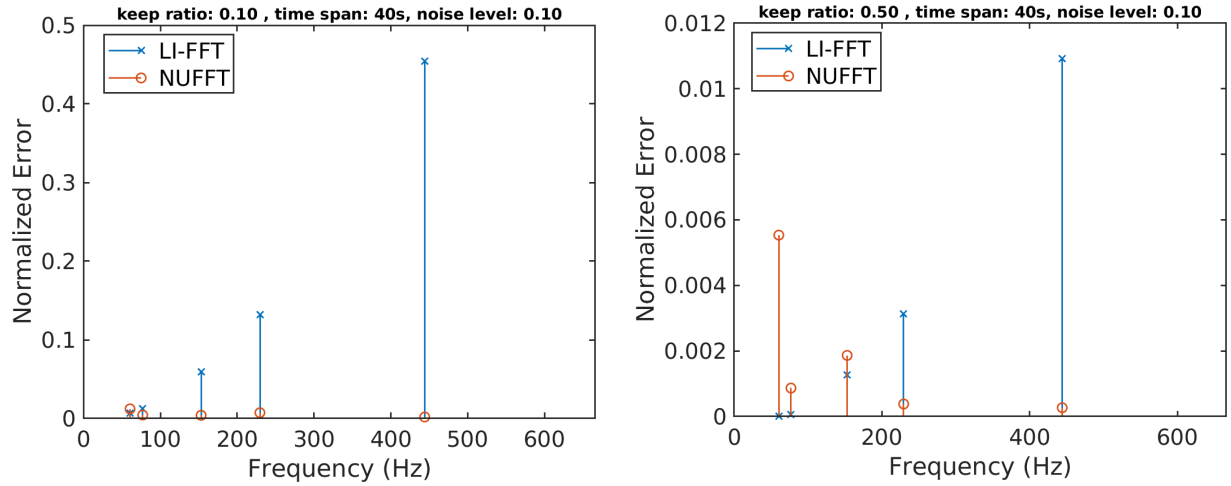



Figure 42: *FFT of the original data.*




FFT using linear interpolation (b) FFT using NUFFT

Figure 43: *FFT of the decimated signal with keep ratio of 0.1.*




(a) keep ratio of 0.1 (b) keep ratio of 0.5

Figure 44: *Comparison of FFT of the decimated signal with keep ratio of 0.1 and 0.5.*

It should be noted that the time difference between consecutive samples does not exceed a certain threshold, specifically one-quarter of the signal period. Among the techniques considered, linear interpolation emerges as the most efficient.

### 11.1.1. Interpolation artifacts

The python numpy interpolator, initially used for resampling our raw vibration data to a continuous sample rate, introduced an artifact that typically appears as a slow oscillation in the overall signal magnitude. This artifact usually reaches its lowest point near the middle of the dataset. To highlight the difference, the linear interpolated signal is overlaid directly onto the zero order hold interpolated signal in Figure 45. The cause of this artifact remains uncertain. It is worth noting that employing a similar linear interpolation function in other libraries such as Matlab or SciPy does not result in the same effect.

Additionally, using numpy's zero-order hold interpolation also does not produce the artifact. It is possible that the discrepancy arises from the way numpy's interpolation handles nonlinear data distribution or outliers, or there may be a variation in the way a parameter is initialized. However, regardless of the underlying cause, addressing the issue was dealt with by switching to the SciPy interpolation routines, which have been implemented in the processing pipeline. Reanalyzing the previous datasets with this corrected approach yielded only slight improvements in average detection accuracy, but removing this artifact is expected to significantly benefit the testing of other processing techniques and eliminate a major confounding factor.

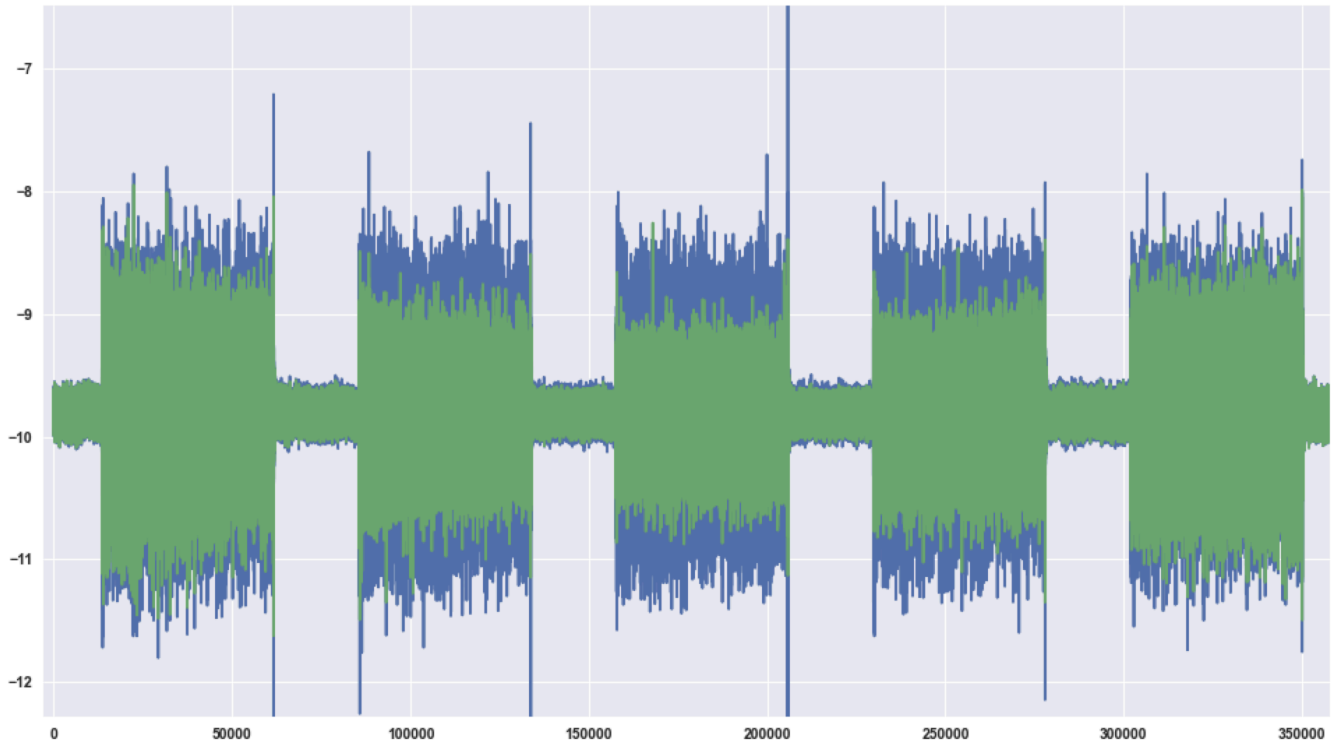



Figure 45: *Zero order hold interpolated data (blue) vs linear interpolated (green). The linear interpolation introduces a significant 'sag' in amplitude across this dataset.*

## 11.2. Method for 2D Scaling

In this section, we explore scaling approaches intended to remove the effects of changing rotation speed from damage detection algorithms. This scaling is referred to here as "2D" because the scaling is applied in time and amplitude.

Consider measurements on a system such that the measured quantity $f(t)$ is given by

$$f(t) = \alpha\phi(\beta t)$$

It is assumed that the function $\phi$ does not change unless the system is changed. However, $\alpha$ and $\beta$ may change even if the system has not changed. Changing is an amplitude scaling while changing $\beta$ is a time scaling. Consider measurements $f_1(t)$ at State 1 given by

$$f_1(t) = \alpha_1 \phi(\beta_1 t).$$

and measurements $f_2(t)$ taken at State 2. For the moment, assume that the system has not changed. Then

$$f_2(t) = \alpha_2 \phi(\beta_2 t).$$

We construct a function $g(t)$ that scales $f_2(t)$ in time and amplitude according to

$$g(t) = a\, f_2(bt).$$

We shall refer to $a$ and $b$ as scaling parameters. If the system has not changed, then

$$f_1(t) = g(t)$$

if the scaling parameters are

$$a = \alpha_1/\alpha_2,$$
$$b = \beta_1/\beta_2.$$

However, if the system has changed then $\phi$ has changed and there is no choice of $a$ and $b$ will bring $g(t)$ into agreement with $f_1(t)$. By using a two-dimensional scaling, one can only bring $f_1(t)$ into agreement with $g(t)$ if the system has not changed.

This result inspires a simple algorithm that solves the following problem:

**Given** Measurements $f_1(t)$ and $f_2(t)$

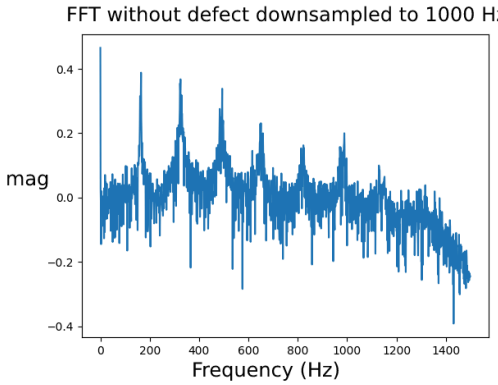
**Find** If the system has changed

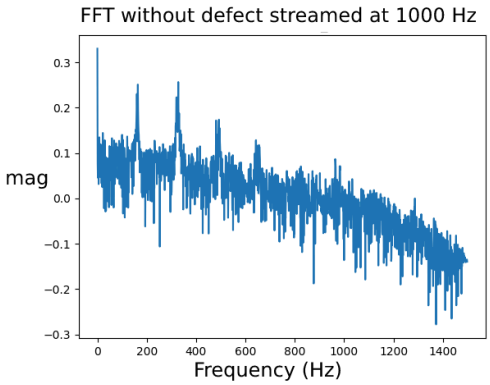
The scaling constants $a$ and $b$ are found by minimizing the Root Mean Square (RMS) error between $f_1(t)$ and $g(t)$.

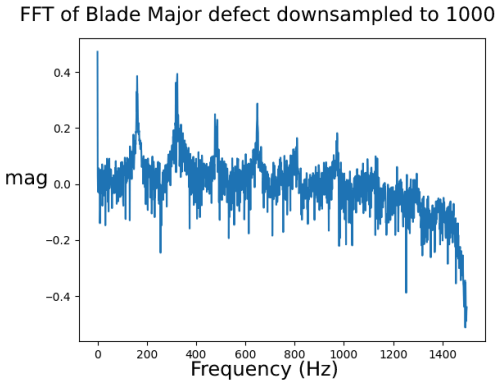
This minimization problem can be solved in many ways, such as a brute force scan over ranges of $a$ and $b$. If values of $a$ and $b$ may be found such that $\varepsilon = 0$, then the system did not change. In practice, there will be noise and measurement errors that will enter the picture, so that $\varepsilon$ might not be exactly zero but very small. Therefore, with experience one can find a range of $\varepsilon$ that will lead to the conclusion that the system did not change.

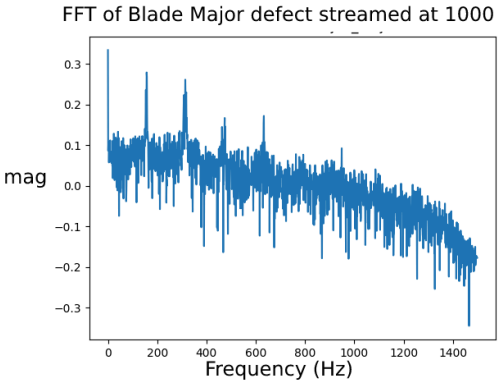
## 11.3. Sample rate differences with streamed data

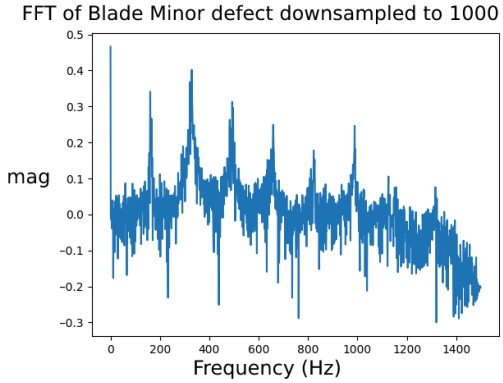
During streaming of sensor data, an average sample rate drop from 1600 Hz to 1000 Hz was observed. This reduction occurred only when the system was configured for live streaming of samples across its 1.5Mbaud serial link. The following graphs demonstrate the differences in the frequency spectrum of each defect when downsampled from 1600 Hz to 1000 Hz compared to being streamed at 1000 Hz. Notably, the higher frequency peaks were unexpectedly flattened in the streamed case. The underlying cause of this discrepancy has not been identified, however detection models trained on the live streamed data were able to detect defects with high (>95%) accuracy on live streamed test data.

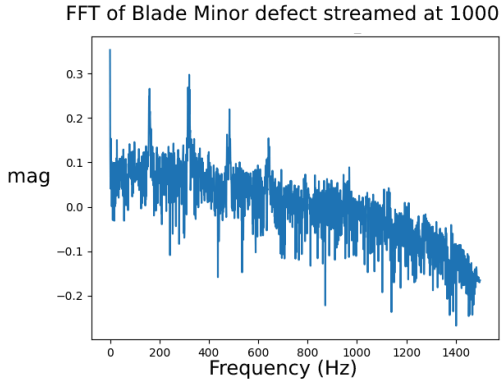

FFT without defect downsampled to 1000 Hz
mag
Frequency (Hz)


FFT without defect streamed at 1000 Hz
mag
Frequency (Hz)


FFT of Blade Major defect downsampled to 1000 Hz
mag
Frequency (Hz)


FFT of Blade Major defect streamed at 1000 Hz
mag
Frequency (Hz)


FFT of Blade Minor defect downsampled to 1000 Hz
mag
Frequency (Hz)


FFT of Blade Minor defect streamed at 1000 Hz
mag
Frequency (Hz)

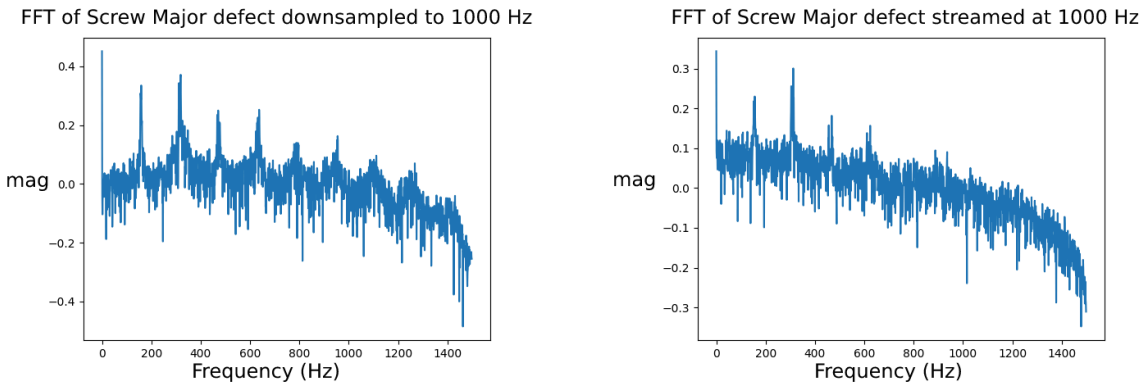



Figure 46: *Comparison of frequency spectrum of data downsampled to 1000 Hz vs streamed at 1000 Hz*